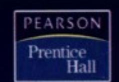
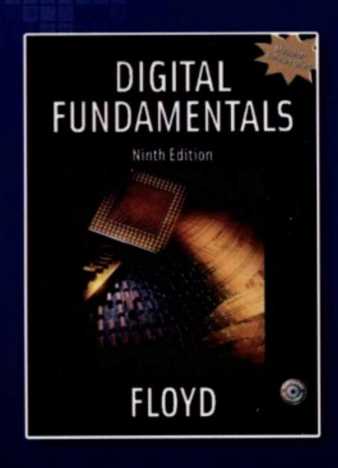
国外电子与通信教材系列



改编版

数字电子技术
（第九版）

Digital Fundamentals, Ninth Edition



[美] Thomas L. Floyd 著
余 璆 等译



http://www.phei.com.cn

数字电子技术（第九版）

改编版

Digital Fundamentals, Ninth Edition

本书是关于数字电子技术的经典教材，内容涉及数字电子技术的基本概念、数制、逻辑门、布尔代数和逻辑化简、组合逻辑分析、组合逻辑的作用、计数器、移位寄存器、存储器、可编程逻辑与软件、集成电路技术等。全书的特色在于示例与习题丰富、图解清晰、语言流畅、深入浅出、证明严谨。本书强调了应用，并帮助读者培养职业生涯所需的实用技能。

本书可作为高等院校电子信息类专业本科生的教材，也可供相关技术、科研管理人员使用，或作为继续教育的参考书。

本书支持网站：www.prenhall.com/floyd

本书英文版



ISBN: 7-121-02632-5
定价：45.00元

余璆：副教授，上海工程技术大学，从事电子技术的教学工作多年。

PEARSON
Prentice Hall



责任编辑：冯小贝
责任美编：喻 晓





定价：45.00 元

国外电子与通信教材系列

数字电子技术

(第九版)

(改编版)

Digital Fundamentals

Ninth Edition

[美] Thomas L. Floyd 著

余 瓈 等译

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内容简介

本书是关于数字电子技术的经典教材，内容涉及数字电子技术的基本概念、数制、逻辑门、布尔代数和逻辑化简、组合逻辑分析、组合逻辑的作用、计数器、移位寄存器、存储器、可编程逻辑与软件、集成电路技术等。全书的特色在于示例与习题丰富、图解清晰、语言流畅、写作风格简约。

本书可作为高等院校电子信息类专业本科生的教材，也可供相关技术、科研管理人员使用，或作为继续教育的参考书。

图书在版编目（CIP）数据

数字电子技术：第9版：改编版/（美）弗洛伊德（Floyd, T. L.）著，余璆 等译

北京：电子工业出版社，2008.5

（国外电子与通信教材系列）

书名原文：Digital Fundamentals, Ninth Edition

ISBN 978-7-121-06447-0

Ⅰ.数… Ⅱ.①弗… ②余… Ⅲ.数字电路－电子技术－高等学校－教材 Ⅳ.TN79

中国版本图书馆CIP数据核字（2008）第057072号

责任编辑：冯小贝　　特约编辑：朱　巍

印　　刷：北京市顺义兴华印刷厂

装　　订：三河市双峰印刷装订有限公司

出版发行：电子工业出版社

　　　　　北京市海淀区万寿路173信箱　邮编：100036

开　　本：787×1092　1/16　　印张：30　　字数：768千字

印　　次：2008年5月第1次印刷

定　　价：45.00元

凡所购买电子工业出版社的图书有缺损问题，请向购买书店调换；若书店售缺，请与本社发行部联系。联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

序

2001年7月间，电子工业出版社的领导同志邀请各高校十几位通信领域方面的老师，商量引进国外教材问题。与会同志对出版社提出的计划十分赞同，大家认为，这对我国通信事业、特别是对高等院校通信学科的教学工作会很有好处。

教材建设是高校教学建设的主要内容之一。编写、出版一本好的教材，意味着开设了一门好的课程，甚至可能预示着一个崭新学科的诞生。20世纪40年代MIT林肯实验室出版的一套28本雷达丛书，对近代电子学科、特别是对雷达技术的推动作用，就是一个很好的例子。

我国领导部门对教材建设一直非常重视。20世纪80年代，在原教委教材编审委员会的领导下，汇集了高等院校几百位富有教学经验的专家，编写、出版了一大批教材；很多院校还根据学校的特点和需要，陆续编写了大量的讲义和参考书。这些教材对高校的教学工作发挥了极好的作用。近年来，随着教学改革不断深入和科学技术的飞速进步，有的教材内容已比较陈旧、落后，难以适应教学的要求，特别是在电子学和通信技术发展神速、可以讲是日新月异的今天，如何适应这种情况，更是一个必须认真考虑的问题。解决这个问题，除了依靠高校的老师和专家撰写新的符合要求的教科书外，引进和出版一些国外优秀电子与通信教材，尤其是有选择地引进一批英文原版教材，是会有好处的。

一年多来，电子工业出版社为此做了很多工作。他们成立了一个“国外电子与通信教材系列”项目组，选派了富有经验的业务骨干负责有关工作，收集了230余种通信教材和参考书的详细资料，调来了100余种原版教材样书，依靠由20余位专家组成的出版委员会，从中精选了40多种，内容丰富，覆盖了电路理论与应用、信号与系统、数字信号处理、微电子、通信系统、电磁场与微波等方面，既可作为通信专业本科生和研究生的教学用书，也可作为有关专业人员的参考材料。此外，这批教材，有的翻译为中文，还有部分教材直接影印出版，以供教师用英语直接授课。希望这些教材的引进和出版对高校通信教学和教材改革能起一定作用。

在这里，我还要感谢参加工作的各位教授、专家、老师与参加翻译、编辑和出版的同志们。各位专家认真负责、严谨细致、不辞辛劳、不怕琐碎和精益求精的态度，充分体现了中国教育工作者和出版工作者的良好美德。

随着我国经济建设的发展和科学技术的不断进步，对高校教学工作会不断提出新的要求和希望。我想，无论如何，要做好引进国外教材的工作，一定要联系我国的实际。教材和学术专著不同，既要注意科学性、学术性，也要重视可读性，要深入浅出，便于读者自学；引进的教材要适应高校教学改革的需要，针对目前一些教材内容较为陈旧的问题，有目的地引进一些先进的和正在发展中的交叉学科的参考书；要与国内出版的教材相配套，安排好出版英文原版教材和翻译教材的比例。我们努力使这套教材能尽量满足上述要求，希望它们能放在学生们的课桌上，发挥一定的作用。

最后，预祝“国外电子与通信教材系列”项目取得成功，为我国电子与通信教学和通信产业的发展培土施肥。也恳切希望读者能对这些书籍的不足之处、特别是翻译中存在的问题，提出意见和建议，以便再版时更正。

吴佑寿
中国工程院院士、清华大学教授
“国外电子与通信教材系列”出版委员会主任

出 版 说 明

进入21世纪以来，我国信息产业在生产和科研方面都大大加快了发展速度，并已成为国民经济发展的支柱产业之一。但是，与世界上其他信息产业发达的国家相比，我国在技术开发、教育培训等方面都还存在着较大的差距。特别是在加入WTO后的今天，我国信息产业面临着国外竞争对手的严峻挑战。

作为我国信息产业的专业科技出版社，我们始终关注着全球电子信息技术的发展方向，始终把引进国外优秀电子与通信信息技术教材和专业书籍放在我们工作的重要位置上。在2000年至2001年间，我社先后从世界著名出版公司引进出版了40余种教材，形成了一套“国外计算机科学教材系列”，在全国高校以及科研部门中受到了欢迎和好评，得到了计算机领域的广大教师与科研工作者的充分肯定。

引进和出版一些国外优秀电子与通信教材，尤其是有选择地引进一批英文原版教材，将有助于我国信息产业培养具有国际竞争能力的技术人才，也将有助于我国国内在电子与通信教学工作中掌握和跟踪国际发展水平。根据国内信息产业的现状、教育部《关于“十五”期间普通高等教育教材建设与改革的意见》的指示精神以及高等院校老师们反映的各种意见，我们决定引进“国外电子与通信教材系列”，并随后开展了大量准备工作。此次引进的国外电子与通信教材均来自国际著名出版商，其中影印教材约占一半。教材内容涉及的学科方向包括电路理论与应用、信号与系统、数字信号处理、微电子、通信系统、电磁场与微波等，其中既有本科专业课程教材，也有研究生课程教材，以适应不同院系、不同专业、不同层次的师生对教材的需求，广大师生可自由选择和自由组合使用。我们还将与国外出版商一起，陆续推出一些教材的教学支持资料，为授课教师提供帮助。

此外，“国外电子与通信教材系列”的引进和出版工作得到了教育部高等教育司的大力支持和帮助，其中的部分引进教材已通过“教育部高等学校电子信息科学与工程类专业教学指导委员会”的审核，并得到教育部高等教育司的批准，纳入了“教育部高等教育司推荐——国外优秀信息科学与技术系列教学用书”。

为做好该系列教材的翻译工作，我们聘请了清华大学、北京大学、北京邮电大学、南京邮电大学、东南大学、西安交通大学、天津大学、西安电子科技大学、电子科技大学、中山大学、哈尔滨工业大学、西南交通大学等著名高校的教授和骨干教师参与教材的翻译和审校工作。许多教授在国内电子与通信专业领域享有较高的声望，具有丰富的教学经验，他们的渊博学识从根本上保证了教材的翻译质量和专业学术方面的严格与准确。我们在此对他们的辛勤工作与贡献表示衷心的感谢。此外，对于编辑的选择，我们达到了专业对口；对于从英文原书中发现的错误，我们通过与作者联络、从网上下载勘误表等方式，逐一进行了修订；同时，我们对审校、排版、印制质量进行了严格把关。

今后，我们将进一步加强同各高校教师的密切关系，努力引进更多的国外优秀教材和教学参考书，为我国电子与通信教材达到世界先进水平而努力。由于我们对国内外电子与通信教育的发展仍存在一些认识上的不足，在选题、翻译、出版等方面的工作中还有许多需要改进的地方，恳请广大师生和读者提出批评及建议。

电子工业出版社

教材出版委员会

译 者 序

数字电子技术是工科电类专业的一门专业基础课，尤其是对自动化和计算机专业而言，这门课对后续课程的进一步学习非常重要。又由于计算机、集成电路的不断发展和数字电路本身层出不尽的应用，使得这门学科不断地得到新的充实。因此有必要翻译引进这方面的国外优秀教材。

本书译自电子工业出版社出版的《数字电子技术(第九版)(英文改编版)》一书。改编版取自 Thomas L. Floyd 所著的 *Digital Fundamentals, Ninth Edition*。原著的内容和结构与国内的数字电子技术教材比较吻合，在这一基础上考虑到内容的精简，以使得它更符合国内教学的要求和课时量，因此推出了英文改编版，作为一本双语教材供学校选用。

这次又在英文改编版的基础上进行了翻译，其一是希望把国外的数字电子技术教材介绍进来，因为它的教学内容和国内的教材既有相同之处，也有它的丰富特性。比如原理的讲解较为详细和清晰，一些章节带有实例或综合举例；同时结合数字逻辑电路，介绍一些计算机小知识。第二个目的是此书可以作为双语教材《数字电子技术(第九版)(英文改编版)》的参考书，希望能够帮助教师和学生完成双语教学的教与学的任务。当然，此书也可以作为应用数字电子技术的相关人员参考。

由于译者的水平有限，一定会有一些错误和不妥之处，恳请读者不吝指教。

目　　录

第1章 数字概念

章节提纲

1.1 数字量和模拟量

1.2 二进制数、逻辑电平和数字波形

1.1 数字量和模拟量

电子电路可以分为两大类:数字电路和模拟电路。数字电路涉及离散的数量,模拟电路涉及连续的数量。尽管在本书中将学习数字电路,但也需要知道一些有关模拟电路的知识,因为很多应用场合都需要两者的技术;而且,模拟电路和数字电路之间的接口也是很重要的。

学完本节以后,应该能够

- 定义模拟量
- 定义数字量
- 指出数字量与模拟量的区别
- 陈述数字量与模拟量相比的优点
- 给出数字量和模拟量如何用于电子领域的例子

模拟量具有连续的数值,数字量具有离散的数值。自然界中大多数可以测量的事物都以模拟量的形式出现。例如,空气温度在一个连续的范围内变化。在给定的一天里,温度不会立即从70°F上升到71°F;这中间经历了无数个温度值。如果绘制一个典型的夏季温度图,那么可以得到一个平滑和连续的曲线,即类似于图1.1的曲线。其他模拟量的例子包括时间、压力、距离和声音。

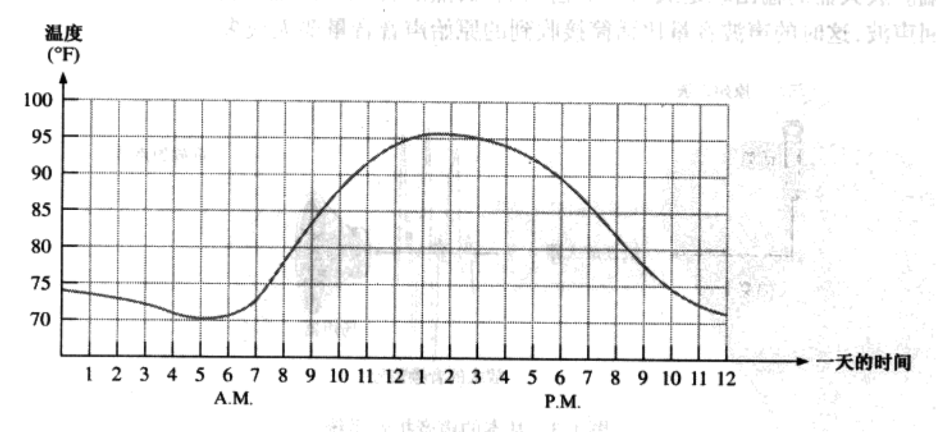


图1.1 模拟量图(温度/时间)

相对于一个连续的温度图,假设每小时测量一次温度。现在有一个24小时内每隔一小时采样测量到的离散温度值,如图1.2所示。这样就可以有效地将模拟量转换成数字量的形式,即用一个个数字码对应于每个采样到的温度值。注意,图1.2本身并不是模拟量的数字表示。

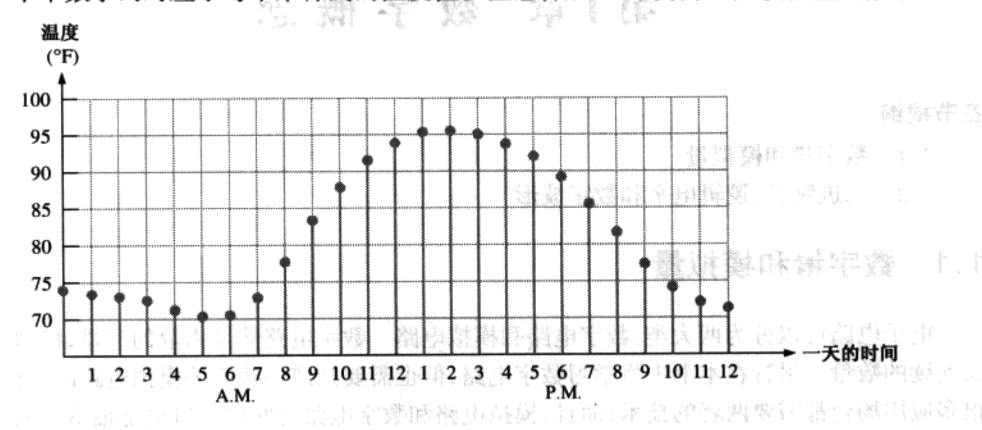


图1.2 图1.1的模拟量样本值的表示法。每个值由点表示,它可以由一些0和1组成的数字码表示

数字量的优点 在电子学应用方面,数字表示法与模拟量表示法相比有一定的优势。其一,数字数据和模拟数据相比,前者在处理和传输方面更有效、更可靠。其二,数字数据在需要保存时,更显示了它的优越性。例如,转换成数字形式的音乐,比对应的模拟形式更简单,并且在复制时更精确,声音更清晰。噪声(不需要的电压波动)几乎不会影响数字数据,但会影响模拟信号。

1.1.1 模拟电子系统

扩音系统用于把声音放大从而让更多的听众听到,这是模拟电子应用的一个简单例子。图1.3给出了自然界中的模拟量——声波,它被话筒接收,并转换为较弱的模拟电压,称为音频信号。这个电压随着声音音量的大小和频率的变化而连续变化,随即加入到线性放大器的输入端。放大器的输出就是放大了的输入电压,然后传入扬声器。扬声器将放大的音频信号再变回声波,这时的声波音量比话筒接收到的原始声音音量要大很多。

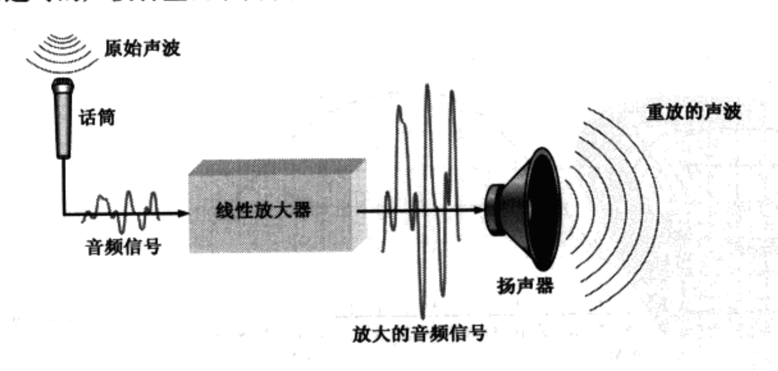


图1.3 基本的声音扩音系统

1.1.2 使用数字方法与模拟方法的系统

光盘(CD)播放器是一个同时使用数字电路和模拟电路的系统。图1.4的简单框图给出了它的基本结构原理。

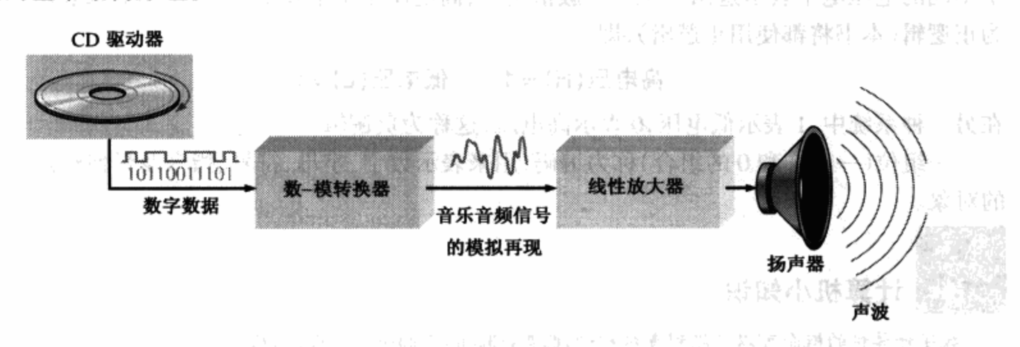


图1.4 光盘播放器基本结构框图,这里只画出了一个通道

数字格式的音乐存储在光盘上。激光二极管光学系统接收旋转光盘上的数字数据,然后将其传送到数-模转换器(DAC)。数-模转换器将这些数字数据转换成模拟信号,即原来音乐电子意义上的再现。线性放大器把模拟信号放大并传送到扬声器,以供欣赏。当把音乐存储在一张CD上时,处理过程基本上和上述过程相反,这时使用模-数转换器(ADC)。

1.2 二进制数、逻辑电平和数字波形

数字电子学涉及电路和系统,其中仅有两种可能的状态。这两种状态使用两个不同的电压电平表示,即高电平(HIGH)和低电平(LOW)。这两种状态也可以表示为两个不同的电流电平、CD或DVD上的凹下和隆起部分等。在计算机这样的计数系统中,这两种状态的组合称为编码,用来表示数字、符号、字母和其他类型的信息。这种具有两种状态的计数系统称为二进制系统,它的两个数为0和1。一个二进制数称为一个位。

学习完本节以后,应当能够

- 定义二进制
- 定义位
- 确认二进制系统中的位
- 说明电压电平如何表示位
- 解释数字电路是如何表示电压电平的
- 描述脉冲的一般特性
- 确定脉冲的幅度、上升时间、下降时间及宽度
- 判断并描述数字波形的特性
- 确定数字波形的幅度、周期、频率和占空比
- 解释什么是时序图及其用途
- 解释串行和并行数据传输,并说出它们各自的优点和缺点

1.2.1 二进制数

二进制系统中的两个数 1 和 0 称为位(bit,即 binary digit 的缩写),在数字电路里,使用两个不同的电压电平表示这两个位。一般情况下,高电压用 1 来表示,低电压用 0 来表示。这称为正逻辑(本书将都使用正逻辑),即

$$高电压(H)=1 \qquad 低电压(L)=0$$

在另一种系统中,1 表示低电压,0 表示高电压,这称为负逻辑。

一组位(一些 1 和 0 的组合)称为编码,用来表示数字、字母、符号、指令及任何给定应用中的对象。



计算机小知识

数字计算机的概念可以追溯到查尔斯·巴巴基(Charles Babbage)时代,他在 19 世纪 30 年代发明了一台原始的机械式计算器。约翰·阿特纳索夫(John Atanasoff)于 1939 年首先在数字计算机中使用了电子处理方法。1946 年,第一台使用真空电子管电路并被称为 ENIAC 的数字计算机诞生。尽管它的体积占据了整个房间,但是 ENIAC 的计算能力还不如我们现在的手持计算器。

1.2.2 逻辑电平

用来表示 1 和 0 的电压称为逻辑电平。理想情况下,一个电平表示高电压,另一个电平表示低电压。在实际数字电路中,这个高电压可以是指定的最小值和最大值之间的任意值。同样,低电压也可以是指定的最小值和最大值之间的任意值。在指定的高电平范围和低电平范围之间不能有重叠。

图 1.5 给出了数字电路中高电平和低电平通常的变化范围。变量 $V_{H(max)}$ 表示高电平的最大值,变量 $V_{H(min)}$ 表示高电平的最小值。$V_{L(max)}$ 表示低电平的最大值,$V_{L(min)}$ 表示低电平的最小值。在正常工作中,$V_{L(max)}$ 和 $V_{H(min)}$ 之间的电压值是不可以出现的。因为在此范围内的电压既可以是高电平也可以是低电平,所以这些值不能出现。例如,在 CMOS 数字电路中,高电平值在 2 ~ 3.3 V 之间,低电平值在 0 ~ 0.8 V 之间,也就是说,如果使用 2.5 V,电路将把它看做高电平或二进制 1。如果使用 0.5 V,那么就是低电平或二进制 0。在这种类型的电路中,0.8 ~ 2 V 之间的电平值是不可以出现的。

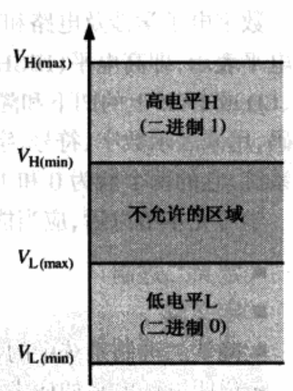


图 1.5 数字电路逻辑电平的电压范围

1.2.3 数字波形

数字波形由不同的电平值组合而成,这些电平值在高低电平或状态之间上下变化。图 1.6(a)给出一个正向脉冲,它从低电平变化到高电平,再从高电平回到低电平。图 1.6(b)给出一个反向脉冲,它从高电平变化到低电平,再从低电平回到高电平。数字波形由这一系列的脉冲组成。

脉冲 如图 1.6 所示，脉冲有两个边沿：在 t_0 时刻首先出现的为前沿，在 t_1 时刻随后出现的为后沿。一个正向脉冲，前沿是上升沿，后沿是下降沿。图 1.6 所示的脉冲是理想状态下的脉冲，因为假设上升沿和下降沿的变化是没有时间差的(瞬间)。在实际情况下这些变化是有时间差的，但是大多数的数字波形可以假定为理想脉冲。

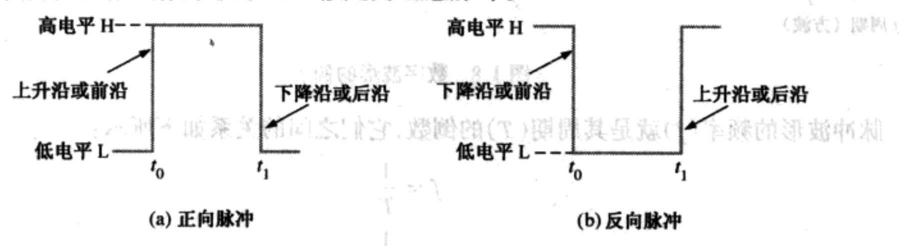


图 1.6 理想的脉冲

图 1.7 给出了一个非理想的脉冲。实际上，所有脉冲或多或少存在这些非理想的特性。通常，杂散电感和电容效应会产生超调量和振荡。杂散电容和电路电阻会产生下垂，形成时间常数不大的 *RC* 电路。

从低电平到高电平所需的时间称之为上升时间 t_r，从高电平到低电平所需的时间称之为下降时间 t_f。在实际运用中，通常测量上升时间是从 10%脉冲幅度(相对于基线的高度)处到 90%脉冲幅度处，测量下降时间则是从 90%幅度处到 10%幅度处。如图 1.7 所示，上升时间和下降时间不包括脉冲顶部和底部的 10%，因为这部分区域的波形是非线性的。脉冲的宽度 t_W 就是脉冲的持续时间，通常把上升沿和下降沿的 50%幅度处的间隙时间定义为脉冲宽度，如图 1.7 所示。

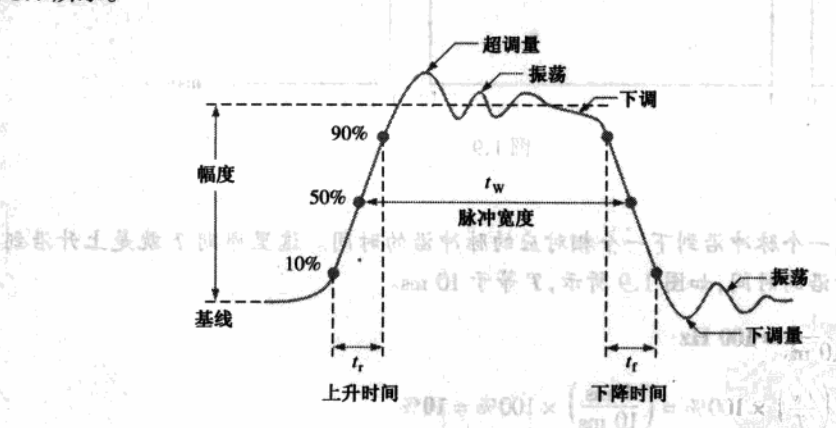


图 1.7 非理想脉冲的特性

波形特性 在计数系统里，遇到的大多数波形都是由一系列的脉冲组成的，有时称为脉冲序列，它们可以分为周期的和非周期的。周期波形就是在一个固定的时间间隔里不断重复自身，这个时间间隔称为周期(T)。频率(f)是重复的速率，测量单位是赫兹(Hz)。而一个非周期性脉冲波形则不会在一个固定的时间间隔里重复，它可能由脉冲宽度不确定的脉冲组成，也有可能由时间间隔不定的脉冲组成，图 1.8 给出两种波形的例子。



图 1.8 数字波形的例子

脉冲波形的频率(f)就是其周期(T)的倒数,它们之间的关系如下所示:

$$f = \frac{1}{T} \tag{1.1}$$

$$T = \frac{1}{f} \tag{1.2}$$

周期数字波形的一个重要特性就是它的占空比(duty cycle),它是脉冲宽度(t_W)和周期(T)的比,可以用百分比来表示:

$$占空比 = \left(\frac{t_W}{T}\right) \times 100\% \tag{1.3}$$

例 1.1 图 1.9 为某周期数字波形的一部分。单位为 ms,试计算:

(a)周期 (b)频率 (c)占空比

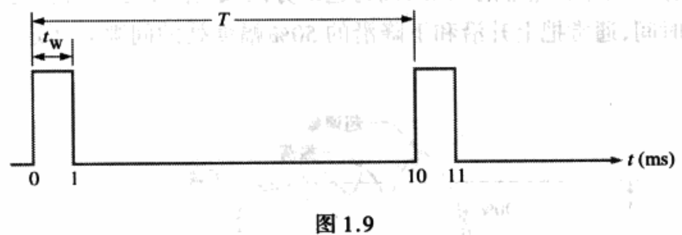


图 1.9

解:

(a)周期是从一个脉冲沿到下一个相对应的脉冲沿的时间。这里周期 T 就是上升沿到下一个上升沿的时间,如图 1.9 所示,T 等于 10 ms。

(b)$f = \frac{1}{T} = \frac{1}{10\ \text{ms}} = \mathbf{100\ Hz}$

(c)占空比 $= \left(\frac{t_W}{T}\right) \times 100\% = \left(\frac{1\ \text{ms}}{10\ \text{ms}}\right) \times 100\% = \mathbf{10\%}$

相关问题: 一个周期数字波形的脉冲宽度为 25 μs,周期为 150 μs,试求出频率和占空比。

1.2.4 数字波形携带二进制信息

计数系统处理的二进制信息以波形的形式出现,它表示一系列的二进制位。当波形为高电平时,表示二进制 1;当波形为低电平时,表示二进制 0。每个位在一个序列里所占的固定时间间隔称为位时间。

时钟 在计数系统中,所有的波形都与一个基本时序波形同步,称之为时钟(clock)。时钟是周期波,每两个脉冲之间的间隔等于一个位时间。

图1.10所示为一个时钟波形的例子。注意,在这种情况下,波形 *A* 的电平变化都是发生在时钟波形的上升沿。在其他情况下,电平的变化发生在时钟的下降沿。在每个位时间之内,波形 *A* 可为高电平也可为低电平。这些高电平和低电平组成了图1.10所示的位序列。若干位组成一组就可以表示一个二进制信息,如数字或字母。而时钟波形本身不携带任何信息。

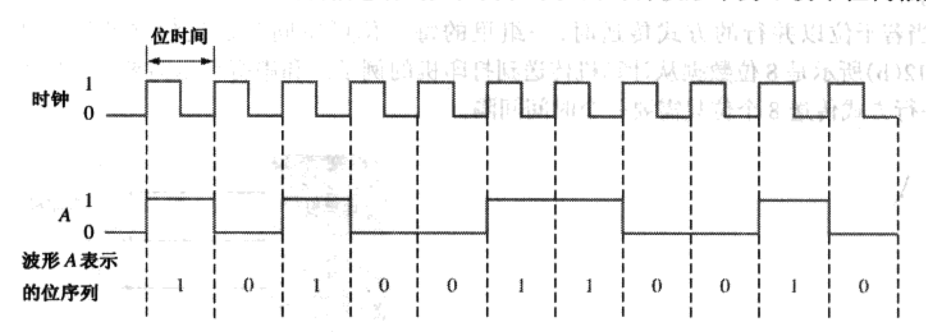


图1.10 时钟波形和位序列表示的波形同步的例子



计算机小知识

计算机的运行速度取决于系统使用的微处理器的类型。计算机速度的定义就是微处理器工作的时钟频率的最大值,例如3.5 GHz。

时序图 时序图就是数字波形的图形,它表示两个或两个以上波形的实际时间关系,还表示波形和波形之间的相互变化关系。图1.11给出了四个波形组成的时序图的例子。从这个时序图中可以确定相互关系。例如,波形 *A*、*B* 和 *C* 仅在位时间7时同为高电平,在位时间7结束时变回到低电平(阴影部分)。

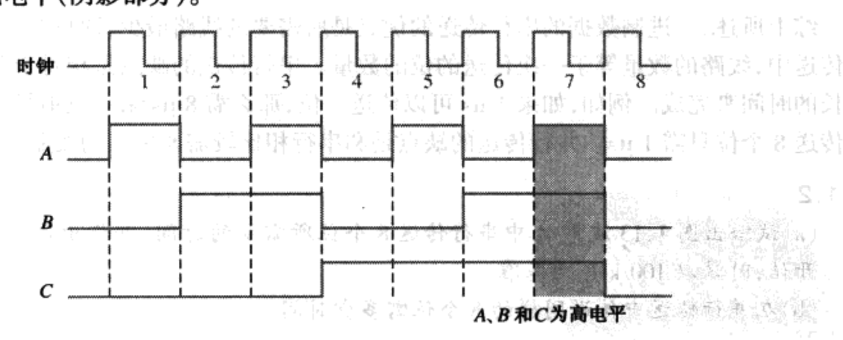


图1.11 时序图的例子

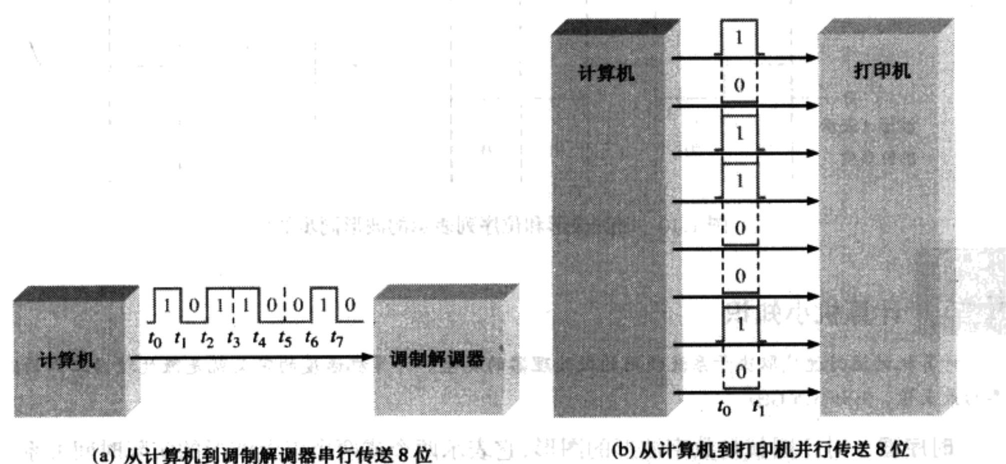
1.2.5 数据传送

数据是指一组可以用来传递某种信息的位。用数字波形表示的二进制数据,必须从一个电路传送到另一个电路,或者从一个系统传送到另一个系统,以实现某个设定的目的。例如,计算机存储器中的数字是以二进制的形式存储的,它必须传送到中央处理器才能实现加法运

算。然后加法运算的结果必须传送到显示器显示并且/或者回送到存储器中。如图 1.12 所示,二进制数据的传送方式有两种:串行和并行。

图 1.12(a)所示为计算机将数据传送到调制解调器的例子,这时位以串行的方式从一个点传送到另一个点,沿着一条导线每次传送一位。从 t_0 到 t_1 这段时间间隔里,送出第一位。从 t_1 到 t_2 这段时间间隔里,送出第二位,以此类推。若要串行输出 8 个位,则需花费 8 个时间间隔。

当若干位以并行的方式传送时,一组里的每一位可以同时通过独立的线路传送。如图 1.12(b)所示是 8 位数据从计算机传送到打印机的例子。和串行传送需要 8 个时间间隔相比,并行方式传送 8 个位只需要一个时间间隔。



(a) 从计算机到调制解调器串行传送 8 位二进制数据。第一个间隔为 t_0 到 t_1

(b) 从计算机到打印机并行传送 8 位二进制数据。开始时间为 t_0

图 1.12 二进制数据串行和并行传送的图例,这里只显示数据线

综上所述,二进制数据的串行传送的优点是所需要的线路最少,即只要一条线路。而在并行传送中,线路的数量等于一次传送的位的数量。串行传送的缺点是相对于并行传送,它需要更长的时间来完成。例如,如果 1 μs 可以传送一位,那么需 8 μs 来完成串行传送 8 个位,但并行传送 8 个位只需 1 μs。并行传送的缺点是和串行相比较需要更多的线路。

例 1.2

(a)试给出图 1.13 波形 A 中串行传送 8 个位所需要的时间,并指出位的顺序。从最左边开始,时钟以 100 kHz 为基准。

(b)在并行传送中传送同样的 8 个位需多少时间?

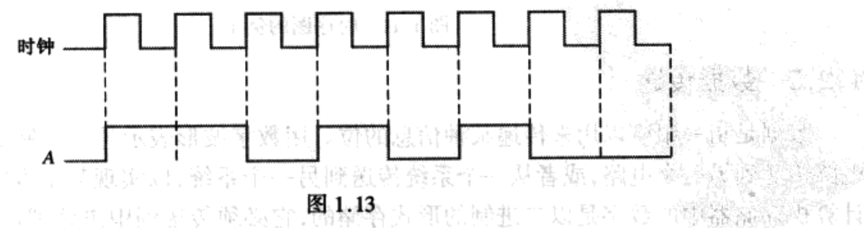


图 1.13

解：

(a) 因为时钟的频率为 100 kHz 时，周期为

$$T = \frac{1}{f} = \frac{1}{100\ \text{kHz}} = 10\ \mu\text{s}$$

传送每位需要 10 μs，传送 8 个位则需要

$$8 \times 10\ \mu\text{s} = \mathbf{80\ \mu s}$$

为了确定每位的顺序，必须确定图 1.13 每个位时间内的波形。如果波形 A 在位时间内为高电平，传送 1。如果波形 A 在位时间内为低电平，则传送 0。位的顺序如图 1.14 所示。从最左边的位最先开始传送。

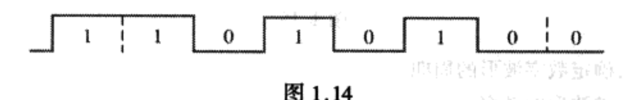


图 1.14

(b) 并行传送 8 个位需要 **10 μs**。

相关问题： 如果二进制数据以 1000 万个位每秒的速度传送，那么并行的 16 个位在 16 条线路上传送需要多少时间？串行传送 16 个位需要多少时间？

自测题 (答案在本章的结尾)

1. 有连续值的量是
   (a)数字量 (b) 模拟量 (c)二进制量 (d)自然量
2. 位的含义是
   (a)少量的数据 (b)一个 1 或一个 0 (c)二进制数 (d)答案(b)和(c)
3. 脉冲前沿位于幅度 10%和 90%之间的时间间隔是
   (a)上升时间 (b)下降时间 (c)脉冲宽度 (d)周期
4. 在一个给定的波形中，每隔 10 ms 出现一个脉冲。则频率为
   (a)1 kHz (b)1 Hz (c)100 Hz (d)10 Hz
5. 在一个给定的数字波形中，其周期为脉冲宽度的两倍，则占空比为
   (a)100% (b)200% (c)50%

习题

1.1 节 数字量与模拟量

1. 数字数据与模拟数据相比的优点是什么？给出两个。
2. 除了温度与声音之外，请给出一个模拟量。

1.2 节 二进制数、逻辑电平和数字波形

3. 确定下列电平的序列，用位(1 和 0)表示：
   (a)高、高、低、高、低、低、低、高
   (b)低、低、低、高、低、高、低、高、低
4. 列出下列以位的序列表示的电平(高和低)的序列：
   (a)1011101 (b)11101001



5. 如图 1.15 给出的脉冲,在图上标出

(a)上升时间　(b)下降时间　(c)脉冲宽度　(d)幅度

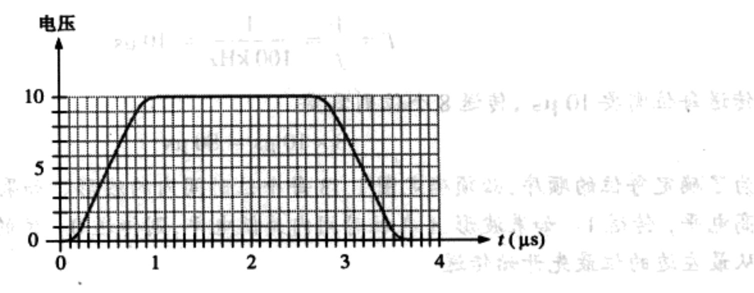


图 1.15

6. 如图 1.16 所示,确定数字波形的周期。
7. 如图 1.16 所示,求波形的频率。
8. 图 1.16 中的脉冲波形是周期性的还是非周期性的?
9. 如图 1.16 所示,求波形的占空比。

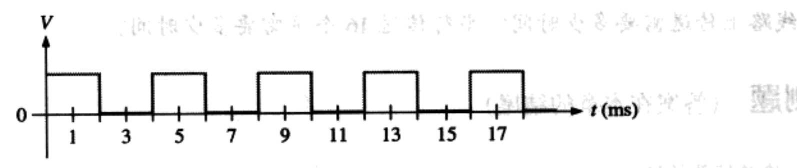


图 1.16

10. 如图 1.17 所示,设位时间为 1 μs,确定波形代表的位序列。
11. 如图 1.17 所示,求串行传送 8 个位所花费的时间? 并行传送 8 个位所花费的时间?

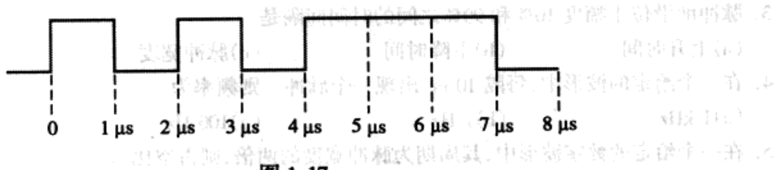


图 1.17

自测题答案

1.(b)　2.(d)　3.(a)　4.(c)　5.(c)

第 2 章　计数系统、运算和编码

章节提纲

2.1　十进制数

我们比较熟悉十进制计数系统，因为每天都在使用十进制数。虽然十进制数很平常，但是它们的加权结构常常没有被认识。这一节我们将回顾十进制数的结构，这将有助于更容易地理解二进制计数系统的结构。二进制计数系统在计算机和数字电子中很重要。

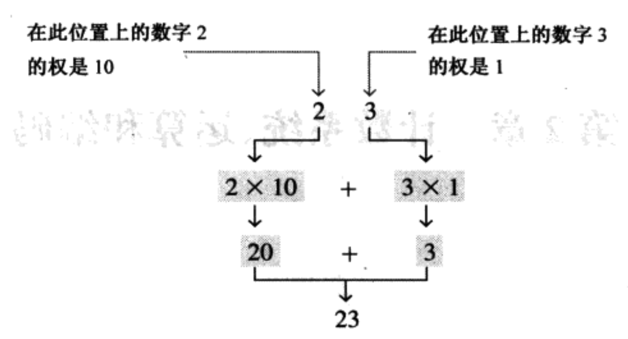
学完本节以后，应该能够

- 解释为什么十进制计数系统是一个有权系统
- 解释 10 的幂如何应用于十进制系统
- 确定十进制数中每一个数字的权

◇ 十进制计数系统具有 10 个数字。

◇ 十进制计数系统的基数为 10。

在十进制计数系统中，10 个数字中的每一个数字(从 0 到 9)都表示某个数量。正如我们所知道的，这 10 个符号(数字)并没有限制为仅表示 10 个不同的数字量，因为可以在数字量相应的位置上分别使用这 10 个不同数字来表示这个数字量的大小。这 10 个数字的每一个最多可以从 0 到 9 表示 10 个数字。如果想要表示一个比 9 大的数字量，可以使用两个或者更多的数字，而每一个数字所处的位置表示了这一位的大小。例如，要表示 23 这个数字量，可以使用(根据它们在该数字量中相应的位置)数字 2 表示 20 这个数字量，用数字 3 表示 3 这个数字量，如下所示：



十进制数中每一个数字所在的位置表示了这个位的大小,称之为权(weight)。整数的权是10的正次幂。从右向左递增,开始于 $10^0 = 1$,

$$\ldots 10^5 \quad 10^4 \quad 10^3 \quad 10^2 \quad 10^1 \quad 10^0$$

对于小数,权是10的负次幂。从左向右递减,开始于 10^{-1},

$$10^2\ 10^1\ 10^0.10^{-1}\ 10^{-2}\ 10^{-3}\ldots$$

↑—— 小数点

◇ 数字的值取决于它在数中的位置。

十进制数的值等于每个数字乘以相应位置上的权之后的和,如例2.1和例2.2所示。

例 2.1 把十进制数 47 表示为各位数字值的求和。

解: 数字 4 的权是 10,由其所在的位置可知为 10^1。数字 7 的权是 1,由其所在的位置可知为 10^0。

$$\begin{aligned} 47 &= (4 \times 10^1) + (7 \times 10^0) \\ &= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7} \end{aligned}$$

相关问题: 确定 939 中每个数字的值。

例 2.2 把十进制数 568.23 表示为各位数字值的求和。

解: 整数 5 的权是 100,即 10^2;整数 6 的权是 10,即 10^1;整数 8 的权是 1,即 10^0;小数 2 的权是 0.1,即 10^{-1};小数 3 的权是 0.01,即 10^{-2}。

$$\begin{aligned} 568.23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0.1) + (3 \times 0.01) \\ &= \mathbf{500} + \mathbf{60} + \mathbf{8} + \mathbf{0.2} + \mathbf{0.03} \end{aligned}$$

相关问题: 确定 67.924 中每个数字的值。

2.2 二进制数

二进制计数系统只是表示数字量的另一种方法。因为它只有两个数字,所以二进制计数系统比十进制计数系统简单。十进制计数系统有10个数字,是以10为基的系统;二进制计数系统有两个数字,并以2为基。两个二进制数字(位)是1和0。二进制数中1或0的位置给出

了它的权，即它在数中所表示的值，就像十进制数字的位置决定该数字的值一样。二进制数的权是基于 2 的幂数。

学完本节以后，应当能够

- 用二进制数计数
- 对于给定数目的位，确定其所能表示的最大十进制数
- 将二进制数转换成十进制数

2.2.1　二进制计数

◇ 二进制计数系统有两个数字(位)。

在学习二进制系统的计数方法时，首先应看一下十进制系统的计数方法。在用完数字之前，可以从 0 开始，依次数到 9。然后从另一个数位开始(左边)，继续从 10 计数到 99。此时，已经用完了两位数字的所有组合，所以需要第三个数位，即从 100 计数到 999。

在进行二进制数计数时，也会发生类似的情况，只是这时只有两个数字，称为位(比特)。开始计数:0,1。此时，已经使用了两个数字，所以加入另一个数位继续计数:10,11。至此，已经使用了两个数字的所有组合，所以需要第 3 个数位。使用 3 个数位，可以继续计数:100,101,110,111。还要继续下去，就需要第 4 个数位，以此类推。从 0 到 15 的二进制计数方法如表 2.1 所示。注意在每一列中 1 和 0 的交替规律。

表 2.1　从 0 到 15 的二进制计数方法

十进制数	二进制数			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

◇ 二进制计数系统的基为 2。

如表 2.1 所示，从 0 计数到 15 需要 4 个数位。一般来说，n 个数位，可以计数到 2^n-1:

$$最大十进制数 = 2^n - 1$$

例如，若有 5 个数位($n=5$)，就可以从 0 计数到 31:

$$2^5 - 1 = 32 - 1 = 31$$

若有 6 个位($n = 6$),就可以从 0 计数到 63:

$$2^6 - 1 = 64 - 1 = 63$$

◇ 一个数中位的值由此数的位置确定。

2.2.2 应用举例

学习二进制数的计数方法,有助于对数字电路如何用来计数有基本的理解。这可以是任意事件的计数,比如从汇编行的程序项条数到计算机的运算次数。举一个简单的例子:计算通过传送带传送到盒子中的网球数,假定向每一个盒子中都传送 9 个球。

图 2.1 中的计数器用来统计来自传感器的脉冲,传感器检测是否有球通过,并在它的 4 个并行输出口上分别输出一系列逻辑电平(数字波形)。每一个逻辑电平的组合都代表了一个 4 位二进数(高电平 = 1,低电平 = 0),如图所示。当译码器接收到这些数字波形时,就会对这个 4 位二进数解码,并在 7 段显示器中把它们转变成相应的十进制数。当计数器达到二进制状态 1001 时,即完成了 9 个网球的计数,显示器显示十进制数 9,同时传送带下移到另外一个盒子。然后计数器回复到 0 状态(0000),计数重新开始。(为了简单起见,这里只给出数字 9 显示一位的情况。)

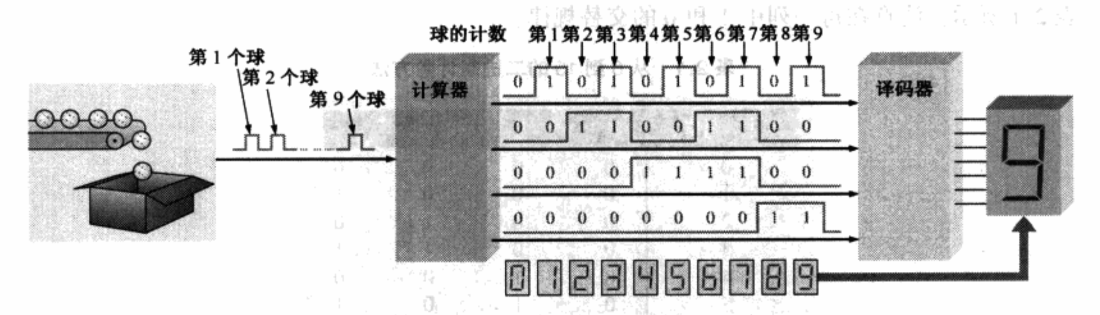


图 2.1 一个简单的二进制计数应用的示意图

2.2.3 二进制数的加权结构

◇ 在一个二进制数中,位的权或值从右向左增加。

二进制数是有权数。在二进制整数中,最右边的位是最低有效位(least significant bit, LSB),并且相应的权是 $2^0 = 1$。权从右向左,每前进一位,2 的幂次增加 1。最左边的位是最高有效位(most significant bit, MSB);二进制数位的大小确定了它的权。

二进制数也可以表示小数,即在二进制小数点的右边添加相应的位就可以了,就像把十进数位添加在十进制小数点右边一样。在二进制小数中,最左边的位是 MSB,其相应的权是 $2^{-1} = 0.5$。小数的权从左向右减少,每位相差 2 的 -1 次幂。

二进制数的加权结构是

$$2^{n-1} \ldots 2^3\, 2^2\, 2^1\, 2^0 . 2^{-1}\, 2^{-2} \ldots 2^{-n}$$

↑——二进制小数点

其中 n 是从二进制小数点开始的位数。因此，在二进制小数点左边的所有的位，其相应的权是 2 的正数幂，就像前面所讨论的整数一样。在二进制小数点右边的所有位，其相应的权是 2 的负数幂，或称小数权。

8 位二进制整数和 6 位二进制小数所对应的 2 的幂次，以及它们对应的十进制权，如表 2.2所示。注意对于 2 的正数幂，权将增大至两倍，而对于 2 的负数幂，权将减半。通过倍增 2 的最高有效正数幂并二等分 2 的最低有效负数幂，就可以很容易地扩展这个表；比如，$2^9 = 512$ 而 $2^{-7} = 0.007\ 812\ 5$。



计算机小知识

计算机使用二进制数来选择存储单元。每个单元都被赋予一个特定的数，称为地址。例如，Pentium 微处理器具有 32 条地址线，可以选择 2^{32}（4 294 967 296）个唯一的单元。

表 2.2　二进制权

2 的正次幂（正整数）									2 的负次幂（小数）					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0.5	0.25	0.125	0.0625	0.03125	0.015625

2.2.4　二进制数－十进制数转换

◇ 把二进制数中所有位是 1 的权相加得到十进制数。

二进制数转换成十进制数，只要把二进制数中的所有位是 1 的权加起来，不考虑所有位是 0 的权。

例 2.3　把二进制整数 1101101 转换为十进制数。

解： 确定每个位是 1 的权，然后把这些权加起来得到十进制数。

$$\text{权：}\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$$
$$\text{二进制数：}\ 1\ 1\ 0\ 1\ 1\ 0\ 1$$
$$1101101 = 2^6 + 2^5 + 2^3 + 2^2 + 2^0$$
$$= 64 + 32 + 8 + 4 + 1 = \mathbf{109}$$

相关问题： 把二进制整数 10010001 转换为十进制数。

例 2.4　将二进制小数 0.1011 转换为十进制数。

解： 确定每个位是 1 的权，然后将这些权加起来得到十进制小数。

$$\text{权：}\ \ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}$$
$$\text{二进制数：}\ 0.1\ \ 0\ \ 1\ \ 1$$
$$0.1011 = 2^{-1} + 2^{-3} + 2^{-4}$$
$$= 0.5 + 0.125 + 0.0625 = \mathbf{0.6875}$$

相关问题： 将二进制数 10.111 转换为十进制数。

2.3 十进制数到二进制数的转换

在上一节中,我们学习了怎样将一个二进制数转换为相应的十进制数。现在学习两种将十进制数转换为二进制数的方法。

学完本节以后,应当能够

- 使用权和的方法将十进制数转换为二进制数
- 使用重复除以 2 的方法将十进制整数转换为二进制数
- 使用重复乘以 2 的方法将十进制小数转换为二进制小数

2.3.1 权和的方法

◇ 想要得到一个给定十进制数的二进制数,只要确定二进制数权的和,它等于相应的十进制数。

求取已知十进制数所对应的二进制数时,一个方法就是确定二进数的一组权,它们的和等于此十进制数。记住二进制权的简单方法是最低位为 1,也就是 2^0,任何一个权乘以 2,就会得到下一个更高位的权;因此,7 个二进制权的列表就是 64,32,16,8,4,2,1,这和上一节中学习的一样。例如,十进制数 9 就可以由二进制权和表示如下:

$$9 = 8 + 1 \quad 或者 \quad 9 = 2^3 + 2^0$$

把 1 放在合适的权的位置上,即 2^3 和 2^0;把 0 放在 2^2 和 2^1 的位置上,就确定了十进制数 9:

2^3 2^2 2^1 2^0

1 0 0 1 十进制数 9 的二进制数表示

例 2.5 将下面的十进制数转换为二进制数:

(a) 12 (b)25 (c)58 (d) 82

解:

(a) $12 = 8 + 4 = 2^3 + 2^2 \longrightarrow$ **1100**

(b) $25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 \longrightarrow$ **11001**

(c) $58 = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1 \longrightarrow$ **111010**

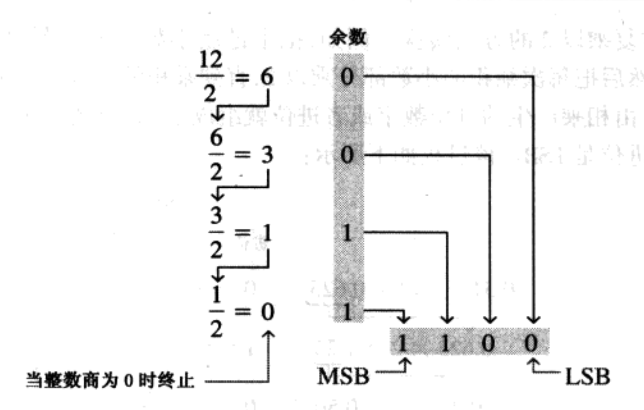
(d) $82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1 \longrightarrow$ **1010010**

相关问题:将十进制数 125 转换为二进制数。

2.3.2 重复除以 2 的方法

◇ 要得到一个给定十进制数的二进制数,可以用 2 除这个十进制数直至商为 0,所有的余数便构成了二进制数。

将十进制整数转换为二进制数的系统方法是重复除以 2 的过程。例如,把十进制数 12 转换为二进制数,首先用 12 除以 2。然后把每次得到的商都除以 2,直到商为 0。每次相除所得到的余数就构成了二进制数。第一个得到的余数是二进制数中的 LSB(最低有效位),最后一个产生的余数是 MSB(最高有效位)。将十进制数 12 转换为二进制数的过程由下列步骤给出。

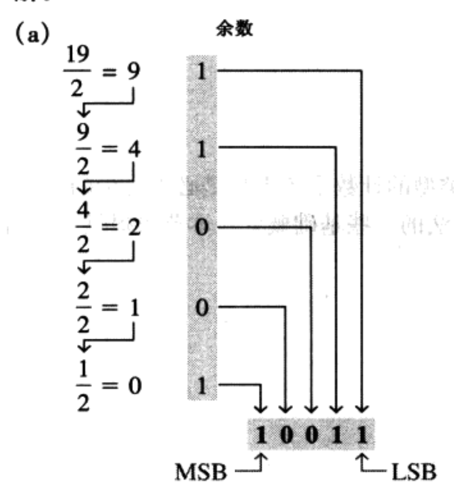


例 2.6 把下面的十进制数转换为二进制数。

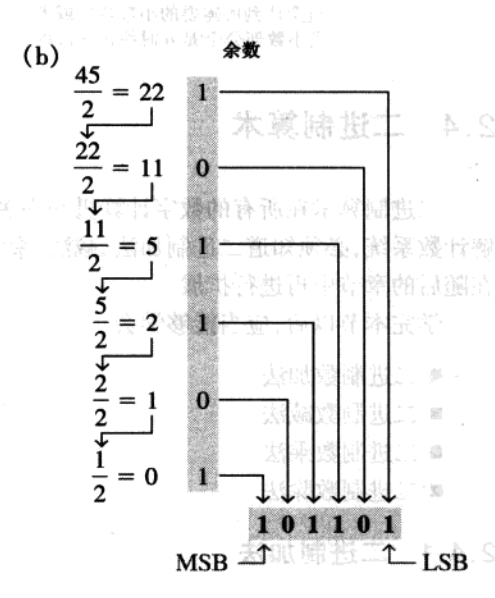
(a) 19 (b) 45

解:

(a)



(b)



相关问题:将十进制数 39 转换为二进制数。

2.3.3 将十进制小数转换为二进制数

例 2.5 和例 2.6 给出了整数的转换,现在来看小数的转换。记住,小数二进制权的方法是最高有效权是 0.5,也就是 2^{-1},任何一个权除以 2,就得到次低位的权;因此 4 个小数的二进制权的序列就是 0.5,0.25,0.125,0.0625。

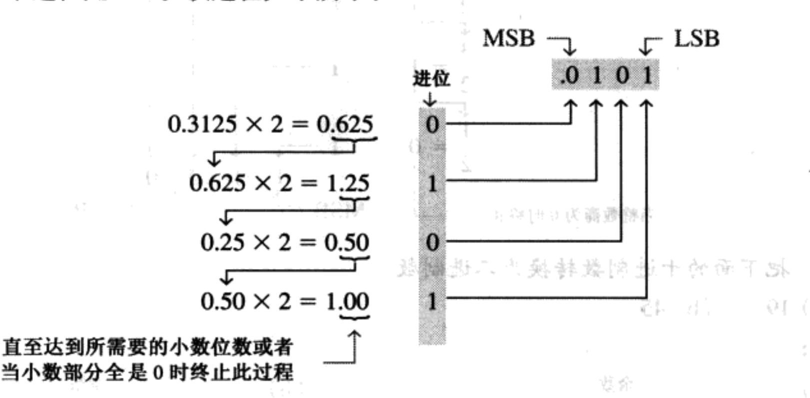
权的和 权和的方法可以应用于十进制小数,例如下面例子:

$$0.625 = 0.5 + 0.125 = 2^{-1} + 2^{-3} = 0.101$$

在 2^{-1}位置上有一个 1,在 2^{-2}位置有一个 0,而在 2^{-3}位置有一个 1。

重复乘以 2 如前所见,可以用重复除以 2 的方法把十进制整数转换为二进制数。二进

制小数则可以用重复乘以 2 的方法转换。例如,把十进制小数 0.3125 转换为二进制数,首先把 0.3125 乘以 2,然后把每次乘积的小数部分乘以 2,直到乘积的小数部分为 0,或者达到了所需要的小数位数。由相乘产生的进位数字或者进位就组成了二进制数。所产生的第一个进位是 MSB,最后一个进位是 LSB。该过程如下所示:



2.4 二进制算术

二进制算术在所有的数字计算机和许多其他类型的计数系统中都是必不可少的。为了理解计数系统,必须知道二进制加法、减法、乘法及除法的一些基础概念。本节将提供一个简介,在随后的章节中再进行扩展。

学完本节以后,应当能够学会

- 二进制数加法
- 二进制数减法
- 二进制数乘法
- 二进制数除法

2.4.1 二进制加法

◇ 记住,在二进制中 1 + 1 = 10,而不是 2。

二进制数(位)加法的四条基本规则如下:

0 + 0 = 0　和为 0,进位是 0
0 + 1 = 1　和为 1,进位是 0
1 + 0 = 1　和为 1,进位是 0
1 + 1 = 10　和为 0,进位是 1

注意:前三条规则产生单个的位,而在第 4 条规则中,两个 1 相加生成二进制的 2(10)。二进制数相加时,最后一种情况在低位的和为 0,而在左边的高位产生了进位 1,如下面的加法运算 11 + 1 所示:

进位　进位

```
  1←   1←
  0  | 1 | 1
 +0  | 0 | 1
 ----------
  1 └0  └0
```

在最右列中,1 + 1 = 0 而在左侧一列中产生进位 1。在中间一列中,1 + 1 + 0 = 0,而在左侧一列中产生进位 1。在最左列中,1 + 0 + 0 = 1。

当存在进位 1 时,就会遇到三个位进行加法的情况(两个数中的位和一个进位)。这种情况如下所示:

进位位

1 + 0 + 0 = 01	和数为 1, 进位为 0
1 + 1 + 0 = 10	和数为 0, 进位为 1
1 + 0 + 1 = 10	和数为 0, 进位为 1
1 + 1 + 1 = 11	和数为 1, 进位为 1

例 2.7　对下列二进制数进行加法运算:

(a) 11 + 11　(b) 100 + 10　(c)111 + 11　(d)110 + 100

解:下面也给出了相应的十进制加法作为对照:

(a)		(b)		(c)		(d)	
11	3	100	4	111	7	110	6
+11	+3	+10	+2	+ 11	+3	+100	+4
110	6	**110**	6	**1010**	10	**1010**	10

相关问题:把 1111 和 1100 相加。

2.4.2　二进制减法

◇ 记住,在二进制中 10 - 1 = 1,而不是 9。

二进制数(位)减法的四条基本规则如下:

0 - 0 = 0

1 - 1 = 0

1 - 0 = 1

10 - 1 = 1　0 - 1 产生借位 1

进行减法运算时,有时必须从左边一列中借位。在二进制中,仅当 0 减去 1 时才需要借位。在此情况下,从左边一列中借来 1,被减的列就会出现 10,这时必须使用上面列出的基本规则中的最后一条。例 2.8 和例 2.9 说明了二进制减法,也给出了相应的十进制减法。

例 2.8　完成下列二进制减法:

(a)11 - 01　(b)11 - 10

解:

(a)		(b)	
11	3	11	3
− 01	− 1	− 10	− 2
10	2	**01**	1

在本例中不需要借位。二进制数 01 和 1 是一样的。

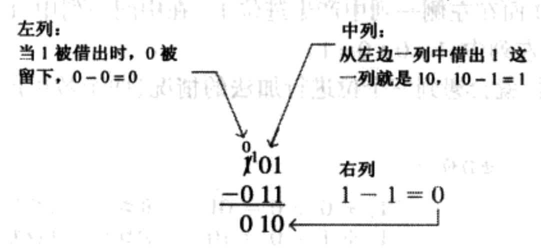
相关问题:从 111 中减去 100。

例 2.9 从 101 中减去 011。

解:

```
  101      5
 -011     -3
  010      2
```

仔细地看一看,在需要借位时,这两个二进制数的相减是怎样进行的。从右边一列开始:



相关问题: 从 110 中减去 101。

2.4.3 二进制乘法

◇ 两位的二进制乘法和十进制数字 1 和 0 的乘法相同。

位相乘的 4 条基本规则如下所示:

$0\times0=0$

$0\times1=0$

$1\times0=0$

$1\times1=1$

二进制乘法和十进制乘法的运算方法是一样的。这涉及部分积的形成,把相继的部分积向左移一位,然后把所有的部分积加起来。例 2.10 说明了这个过程,也给出了相应的十进制乘法作为参照。

例 2.10 运行下面的二进制乘法:

(a) 11×11　　(b) 101×111

解: (a)

```
          11      3
         ×11     ×3
部分积 {  11      9
         +11
        1001
```

(b)

```
          111      7
         ×101     ×5
部分积 {  111     35
          000
        +111
        100011
```

相关问题: 计算 1101×1010。

2.4.4 二进制除法

◇ 只要不超出计算器的计算范围,就可以用它实现二进制数的算术运算。

二进制中的除法遵循和十进制除法一样的过程,例 2.11 给出了相应的十进制除法。

例 2.11 执行下面的二进制除法运算:

(a) $110\div11$　　(b) $110\div10$

解：(a)

$$
\begin{array}{r}
10 \\
11\overline{)110} \\
\underline{11} \\
000
\end{array}
\qquad
\begin{array}{r}
2 \\
3\overline{)6} \\
\underline{6} \\
0
\end{array}
$$

(b)

$$
\begin{array}{r}
11 \\
10\overline{)110} \\
\underline{10} \\
10 \\
\underline{10} \\
00
\end{array}
\qquad
\begin{array}{r}
3 \\
2\overline{)6} \\
\underline{6} \\
0
\end{array}
$$

相关问题：计算 1100 除以 100。

2.5 二进制数的反码和补码

二进制的反码和补码很重要，因为它们允许表达负数。补码算术方法广泛应用于计算机中用以处理负数。

学完本节以后，应当能够

- 把一个二进制数转换为它的反码
- 使用两种方法中的任何一种把二进制数转换为补码

2.5.1 求二进制数的反码

◇ 改变数中的每一位得到反码。

二进制数的反码可以通过把所有的 1 变为 0 及把所有的 0 变为 1 而得到，如下所示：

1 0 1 1 0 0 1 0 二进制数
↓↓↓↓↓↓↓↓
0 1 0 0 1 1 0 1 反码

利用数字电路求二进制数反码的最简单方法是使用并行反相器（“非”电路），如图 2.2 所示，用以转变 8 位二进制数。

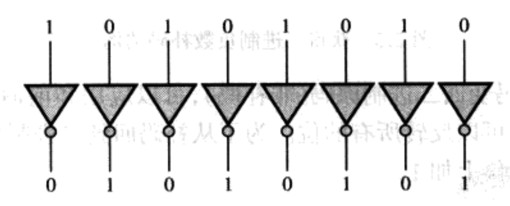


图 2.2 反相器例子，用以获得二进制数的反码

2.5.2 求二进制数的补码

◇ 反码加 1 就可以得到补码。

二进制的补码是通过在反码的 LSB（最低位）上加 1 而获得：

$$补码 = 反码 + 1$$

例 2.12 求 10110010 的补码。

解:

	10110010	二进制数
	01001101	反码
+	1	加 1
	01001110	补码

相关问题:确定 11001011 的补码。

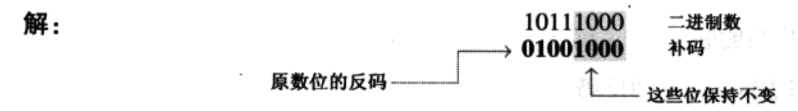
◇ 改变最低有效 1 左边的全部位求得补码。

求二进制数补码的另一种替代方法如下所示:

1. 从右边的 LSB 开始,写下第一次到达 1 的位(也包括这个 1)。
2. 剩下的位求反码。

例 2.13 使用替代方法求 10111000 的补码。

解:



相关问题:求 11000000 的补码。

使用反相器和加法器可以实现求二进制负数的补码,如图 2.3 所示。该图说明了怎样把一个 8 位数转换为它的补码;首先每一个位取反(取得反码),然后使用加法器电路把反码加 1。

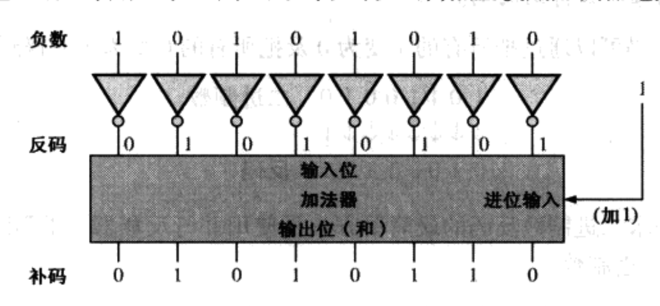


图 2.3 获得二进制负数补码的例子

为了把反码或者补码变回二进制原码(非补码),可以应用和前面所描述的过程。为了从反码回到二进制数原码,可以反转所有的位。为了从补码回到二进制数原码,可以先取得补码的反码,然后在最低有效位上加 1。

2.6 带符号数

计数系统,例如计算机,必须能同时处理正数和负数。带符号的二进制数含有符号和数值信息。符号指明这个数是正数还是负数,而数值就是该数的值。在二进制中,带符号数有三种表达方式:符号 - 数值、反码和补码。在这三种方法中,补码是最重要的,而符号 - 数值很少用。非整数及非常大或非常小的数可以用浮点格式来表示。

学完本节以后,应当能够

■ 用符号 - 数值表示正数和负数

- 用反码表示正数和负数
- 用补码表示正数和负数
- 确定带符号二进制数的十进制值
- 用浮点格式表示二进制数

2.6.1　符号位

带符号二进制数最左边的一位是符号位，用来指明这个数是正数还是负数。

符号位 0 表示正数，1 表示负数。

2.6.2　符号数值的形式

当以符号数值的形式表示带符号二进制数时，最左边的一位是符号位，其余的都是数值位。数值位对正数和负数来讲都是二进制原码(非补码)。例如，十进制数 +25 表示为符号数值的形式就是一个 8 位带符号二进制数，即

00011001

符号位 ↑　↑ 数值位

十进制数 -25 表示为

10011001

注意 +25 和 -25 之间的唯一区别是符号位，因为对于正数和负数来说，数值位都属于二进制原码。

在符号数值形式中，负数和其相应的正数具有相同的数值位，但其符号位为 1 而不是 0。



计算机小知识

计算机在所有的算术运算中都使用补码来表示负整数。原因是减去某个数和加上这个数的补码是一样的。计算机通过按位取反和加 1 来形成补码，使用特殊的指令产生和图 2.3 中的加法器一样的结果。

2.6.3　反码形式

以反码形式表示正数的方法和以符号数值形式表示正数的方法是一致的。但是，负数却是其相应正数的反码。例如，使用八位数字，十进制数 -25 可以表示为 +25(00011001)的反码：

11100110

在反码形式中，负数就是其相应正数的反码。

2.6.4　补码形式

正数的补码形式表示方法和符号数值及反码形式的表示方法是一致的。负数是相应正数的补码。同样，使用八位数字，把十进制数 -25 表示为 +25(00011001)的补码：

11100111

在补码形式中，负数是相应正数的补码。

例 2.14 以符号数值、反码和补码的形式将十进制数 -39 表示为一个八位数。

解:首先,写出 +39 的八位数:

00100111

在符号数值形式中,-39 是通过改变符号位为 1 并保持数值位不变而得到的。这个数是

10100111

在反码形式中,-39 是通过求 +39(00100111)的反码得到的:

11011000

在补码形式中,-39 是通过求 +39(00100111)的补码得到的,如下所示:

```
  11011000   反码
+        1
----------
  11011001   补码
```

相关问题:以符号数值、反码和补码的形式表示 +19 和 -19。

2.6.5 带符号数的十进制值

符号数值 在符号数值形式中,正数和负数的十进制值是由所有数值位为 1 的相应权加起来得到的,不考虑那些为 0 的位。符号通过检查符号位来确定。

例 2.15 确定一个以符号数值表示的带符号二进制数的十进制值:10010101。

解:这 7 个数值位和它们以 2 的幂表示的权如下所示:

2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	0	1

把 1 位置上所对应的权加起来,即 16+4+1=21。
符号位为 1:所以十进制数是 **-21**。

相关问题:确定符号数 01110111 表示的十进制值。

反码 在反码形式中,正数的十进制值是由所有为 1 的数值位相应的权加起来得到的,而不要考虑为 0 的位置。负数的十进制值是通过给符号位的权赋以负值,并把所有为 1 的数值位相应的权加起来,再加上 1 得到的。

例 2.16 确定以反码表示的带符号二进制数的十进制值:

(a) 00010111 (b) 11101000

解:

(a)正数的位和它们以 2 的幂表示的权如下所示:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	1	0	1	1	1

把所有位置为 1 的权加起来,

$$16+4+2+1=\mathbf{+23}$$

(b)负数的位和它们以 2 的幂表示的权如下所示,注意负符号位的权是 -2^7 或者 -128,

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	0	1	0	0	0

把所有位置为 1 的权加起来,

$$-128+64+32+8=-24$$

把结果上加上 1,最终的十进制数是

$$-24+1=\mathbf{-23}$$

相关问题:确定反码数 11101011 的十进制值。

补码　在补码形式中,正数和负数的十进制值,是把所有为 1 的数值位相应的权加起来得到的,而不要考虑 0 的位置。负数中符号位的权被赋以负值。

例 2.17　确定以补码表示的带符号二进制数的十进制值:

(a)01010110　　(b)10101010

解:

(a)正数的位和它们以 2 的幂表示的权如下所示:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	0	1	1	0

把所有位置为 1 的权加起来,

$$64+16+4+2=\mathbf{+86}$$

(b)负数的位和它们以 2 的幂表示的权如下所示,注意负符号位的权是 -2^7 或者 -128,

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	0

把所有位置为 1 的权加起来,

$$-128+32+8+2=\mathbf{-86}$$

相关问题:确定反码数 11010111 的十进制值。

从这些例子中,可以看到为什么用补码形式来表示带符号整数比较好:若转换成十进制数,仅仅需要求权的和,而不用考虑是正数还是负数。对于负数而不是正数,反码形式需要给权的和再加上 1。当然反码形式是不常用的,因为 0 的反码有两种可能的表示方式(0000000 或 11111111)。

2.6.6　带符号整数的表示范围

◇ 二进制数的数值范围取决于数的位数(n)。

使用 8 位数来给出说明,因为在大多数计算机中通常使用 8 位数组,并被赋以特殊的名称——字节(byte)。使用一个字节或者 8 位,可以表示 256 个不同的数。使用两个字节或者 16 位,可以表示 65 536 个不同的数。使用 4 个字节或者 32 位,可以表达 4.295×10^9 个不同的数。求解 n 位的不同组合个数的公式是

$$总组合数 = 2^n$$

对于补码带符号数,n 位数的数值范围是

$$范围 = -(2^{n-1}) 到 +(2^{n-1}-1)$$

其中对于每一个数,都有一个符号位和 $n-1$ 个数字位。例如,使用4位数,并以补码表示的数的范围是从 $-(2^3)=-8$ 到 $2^3-1=+7$。类似地,利用 8 位,就可以表示 -128 到 +127;使用16位,就可以表示 -32 768 到 +32 767;等等。

2.6.7 浮点数

为了表示很大的整数,就会需要多个位。当需要表示的数值同时具有整数和小数部分时(比如 23.5618),就会是一个问题。而基于科学计数法的浮点计数方法系统,就可以表示很大及很小的数,而不用增加位数,当然也可以表示同时具有整数和小数部分的数。

浮点数(也称为实数)由两部分组成再加上一个符号。尾数是浮点数中用以表示数字数值的部分。指数是浮点数中用以表示小数点(或者二进制小数点)要移动的位数。

十进制数的例子将有助于理解浮点数的基本概念。考虑一个十进制数,以整数形式表示为 241 506 800。尾数为 .241 506 8,而指数是 9。当这个整数表示为浮点数时,通过把小数点移动到数字的最左边而将其标准化,使得位数是一个小数,而指数是 10 的幂。这个浮点数写为

$$0.241\,506\,8 \times 10^9$$

对于二进制浮点数来说,其格式由 ANSI/IEEE 标准 754-1985 定义为三种形式:单精度、双精度及扩展精度。除了位数不同之外,它们都具有相同的基本格式。单精度浮点数具有 32 个位,双精度浮点数具有 64 个位,而扩展浮点数具有 80 个位。我们的讨论将限制于单精度浮点数格式。

单精度浮点二进制数 在单精度浮点二进制数的标准格式中,符号位(S)是最左边的位,指数(E)包括接下来的 8 个位,尾数或者小数部分(F)包括剩余的 23 个位,如下所示:

32 位

符号位 (S)	指数 (E)	尾数 (小数, F)
1位	8位	23位



计算机小知识

在 CPU(中央处理器)之外,计算机使用协处理器依靠浮点数来执行复杂的数学计算。目的是释放 CPU 去完成其他任务,从而提高性能。数字协处理器也称为浮点单元(FPU)。

在尾数或者小数部分,一般认为二进制小数点位于 23 位的左边。为了有效起见,尾数部分具有 24 位,因为在任何一个二进制数中最左边(MSB)的位总是为 1。所以,这个 1 被认为位于这里,尽管它并没有占用实际的位置。

指数中的八个位表示的是偏移指数,在实际指数上加上 127 就可以得到。偏移指数的目的是允许表示很大或者很小的数,而不需要一个单独的符号位。偏移指数所允许的实际指数范围是 -126 到 +128。

为了说明二进制数怎样以浮点格式来表示,用 1011010010001 作为一个例子。首先,它可

以表示为 1 加上一个二进制小数，即把二进制小数点向左移动 12 位然后乘以适当的 2 的幂：

$$1011010010001 = 1.011010010001 \times 2^{12}$$

假设这是一个正数，符号位(S)是 0。指数 12 表示为偏移指数的形式：将其加上 127(12 + 127 = 139)。该偏移指数(E)表示为二进制数 10001011。尾数是二进制数的小数部分(F)，即 .011010010001。因为在 2 的幂次表达式中，二进制小数点的左边总是为 1，它并不包含在尾数中。完整的浮点数是

S 符号位	E 指数	F 尾数
0	10001011	01101001000100000000000

接下来，我们来看怎样计算以浮点格式表示的二进制数。确定浮点数数值的一般方法如下面的公式所示：

$$数 = (-1)^S(1 + F)(2^{E-127})$$

为了说明这个公式，考虑下面的浮点二进制数：

S 符号位	E 指数	F 尾数
1	10010001	10001110001000000000000

符号位是 1。偏移指数是 10010001 = 145。应用这个公式，我们得到

$$\begin{aligned} 数 &= (-1)^1(1.10001110001)(2^{145-127}) \\ &= (-1)(1.10001110001)(2^{18}) = -1100011100010000000 \end{aligned}$$

这个浮点二进制数在十进制数中相当于 - 407 688。因为指数可以是 - 126 和 + 128 之间的任意数，所以最大和最小的数都可以表示出来。32 位的浮点数可以代替具有 129 位的二进制整数。因为指数确定了二进制小数点的位置，所以也可以表示同时包含整数和小数部分的数。

浮点数的这种格式有两个例外：数 0.0 由全 0 来表示，而无穷大的数由指数全 1 和尾数全 0 来表示。

例 2.18　把十进制数 3.248×10^4 转换为单精度浮点二进制数。

解：把十进制数转换为二进制数：

$$3.248 \times 10^4 = 32\,480 = (111111011100000)_2 = 1.11111011100000 \times 2^{14}$$

MSB 并不占用位的位置，因为它总是为 1，所以，尾数是 23 位的二进制小数 11111011100000000000000，而偏移量是

$$14 + 127 = 141 = 10001101_2$$

完整的浮点数是

0	10001101	11111011100000000000000

相关问题：求下面浮点二进制数的二进制值：

0 10011000 10000100010100110000000

2.7 带符号数的算术运算

在上一节中,学习了带符号数是如何用三种不同的形式表示的。在这一节中,我们将学习怎样对带符号数进行加、减、乘、除运算。因为表示带符号数的补码形式在计算机和基于微处理器的系统中应用最广泛,所以本节所要讨论的内容仅限于补码算术。如果需要,所讨论的过程可以扩展到其他形式中。

学完本节以后,应当能够

- 学会带符号二进制数的加法
- 解释计算机怎样相加数字串
- 定义溢出
- 学会带符号二进制数的减法
- 使用直接相加方法对带符号二进制数进行乘法运算
- 使用部分积方法对带符号二进制数进行乘法运算
- 学会带符号二进制数的除法

2.7.1 加法

加法中的两个数就是加数和被加数。结果是和。当两个带符号二进制数相加时,有以下4种情况:

1. 两个数都是正的
2. 正数的数值大于负数的数值
3. 负数的数值大于正数的数值
4. 两个数都是负的

这里使用8位带符号数,每次举一种情况作为例子,同时也给出相应的十进制数作为对照。

◇ 两数相加产生一个正数。

两个数都是正的:

```
  00000111       7
+ 00000100     + 4
  00001011      11
```

和是正的,因而是二进制原码(非补码)。

◇ 正数和一个相比而言较小的负数相加产生一个正数。

正数的数值大于负数的数值:

```
                     00001111      15
                   + 11111010    + -6
放弃进位———→  1   00001001       9
```

最后的进位被舍去。和是正的,因而是二进制原码(非补码)。

◇ 一个正数加上一个相比而言较大的负数,或者两个负数相加时,生成一个补码形式的负数。

负数的数值大于正数的数值：

```
  00010000      16
+ 11101000   + -24
  11111000      -8
```

和是负的，所以是补码形式。

两个数都是负的：

```
             11111011     -5
           + 11110111   + -9
放弃进位 → 1 11110010    -14
```

最后的进位被舍去。和是负的，所以是补码形式。

在计算机中，负数是以补码形式保存的，所以正如所见，加法过程是很简单：将两个数加起来，并舍去最后的任何进位。

溢出条件　当两个数加在一起，而表示的和所需的位数超出了这两个数的位数。这时就会发生溢出，并由一个错误符号位指明。溢出仅发生在两个都是正数或者两个都是负数的情况下。下面的 8 位数例子说明了这种情况：

```
  01111101     125
+ 00111010   +  58
  10110111     183
```

符号不正确（指向最高位 1）
大小不正确（指向其余 7 位 0110111）

在这个例子中，和 183 需要 8 个数值位。由于数中只有 7 个数值位（有一位是符号位），这时进位就会进入符号位，从而产生溢出指示。

一次加两个数　现在让我们看看数字串相加的情况，即一次将两个数加起来。首先将前两个数相加，然后两数之和加上第三个数，然后再在此和的基础上加上第四个数，以此类推。这就是计算机中数字串相加的方法。一次相加两个数的方法如例 2.19 所示。

例 2.19　符号数相加：01000100、00011011、00001110 和 00010010。

解：给出了相应的十进制加法，以做对照。

```
   68       01000100
+  27     + 00011011    加前两个数
   95       01011111    第一个和
+  14     + 00001110    加第三个数
  109       01101101    第二个和
+  18     + 00010010    加第三个数
  127       01111111    最后的和
```

相关问题：将 00110011、10111111 和 01100011 这些带符号数相加。

2.7.2　减法

◇ 减法是将减数符号改变后的加法。

减法是加法的一个特例。例如,从+9(被减数)中减去+6(减数)就相当于+9加上-6。基本上,减法运算是改变减数的符号然后加上被减数的运算。减法的结果称为差。

正二进制数或者负二进制数的符号通过求此数的补码而改变。

例如,求取正数00000100(+4)的补码时,就会得到11111100,这就是-4,权和的计算如下:

$$-128+64+32+16+8+4=-4$$

作为另一个例子,当你取负数11101101(-19)的补码时,就会得到00010011,这就是+19,权和的计算如下:

$$16+2+1=19$$

由于减法仅仅是减数符号改变后的加法,所以该过程有如下的表述方式:

要将两个带符号数相减,取减数的补码然后相加即可。舍去最后的任何进位。

例2.20描述了减法过程。

例2.20 完成下面带符号数的减法运算:

(a)00001000-00000011　　(b)00001100-11110111

(c)11100111-00010011　　(d)10001000-11100010

解:像其他例子一样,也给出了相应的十进制减法作为对照。

(a) 在这种情况下,8-3=8+(-3)=5

```
             00001000    被减数(+8)
           + 11111101    减数(-3)的补码
舍去进位 → 1 00000101    差(+5)
```

(b) 在这种情况下,12-(-9)=12+9=21

```
   00001100    被减数(+12)
 + 00001001    减数(+9)的补码
   00010101    差(+21)
```

(c) 在这种情况下,-25-(+19)=-25+(-19)=-44

```
             11100111    被减数(-25)
           + 11101101    减数(-19)的补码
舍去进位 → 1 11010100    差(-44)
```

(d)在这种情况下,-120-(-30)=-120+30=-90

```
   10001000    被减数(-120)
 + 00011110    减数(+30)的补码
   10100110    差(-90)
```

相关问题:01011000减去01000111。

2.7.3 乘法

◇ 乘法相当于一个数加上它本身,相加的次数就是乘数。

乘法中的数分别是被乘数、乘数及积。下面的十进制乘法运算说明了这些数：

```
   8   被乘数
×  3   乘数
  24   积
```

在大多数计算机中，乘法运算是通过加法来完成的。正如所见，减法是由加法器完成的；现在我们看看乘法运算是如何完成的。

直接加法和部分积是使用加法完成乘法的两种基本方法。在直接加法方法中，被乘数自身相加的次数等于乘数。在前面的十进制例子(3×8)中，三个被乘数相加：8+8+8=24。这种方法的缺点是，如果乘数很大运算会变得很冗长。比如，要运算 75×350，必须把 350 自身相加 75 次。顺便说一下，这就是为什么相乘的次数常用来指乘数。

两个二进制数相乘时，这两个数都必须是原码(非补码)。直接加法的方法如例 2.21 所示，每次进行两个二进制数的相加。

例 2.21　乘带符号二进制数：01001101(被乘数)和 00000100(乘数)，使用直接加法的方法。

解：由于两个数都是正数，所以都是原码，积也是正数。乘数的十进制数值为 4，所以被乘数要自身相加 4 次，如下所示：

```
   01001101   第一次
 + 01001101   第二次
   10011010   部分和
 + 01001101   第三次
   11100111   部分和
 + 01001101   第四次
  100110100   乘积
```

相关问题：使用直接加法的方法，把 01100001 乘以 00000110。

部分积方法可能是最常用的一种方法，因为它反映了手工乘法运算。从乘数的最低有效位开始，被乘数被乘以乘数的每一个位。被乘数乘以乘数每一位的结果称为部分积。每一个相继的部分积都向左移动(平移)一位，当所有的部分积都产生时，把它们加起来就得到最后的积。这里有一个十进制例子：

```
     239    被乘数
   × 123    乘数
     717    第一个部分积(3×239)
    478     第二个部分积(2×239)
 + 239      第三个部分积(1×239)
  29 397    最后的积
```

乘法运算的积的符号取决于被乘数和乘数的符号，依据下面两条规则：

- **如果符号相同，积就是正值；**
- **如果符号相异，积就是负值。**

二进制乘法的部分积方法的基本步骤如下所示：

步骤 1：确定被乘数和乘数的符号是相同的还是相异的。这将决定积的符号。

步骤 2: 把所有的负数变为原码(非补码)形式。因为大多数计算机都以补码形式保存负数,将负数转变为原码,需要进行补码运算。

步骤 3: 开始于最低有效乘数位,生成部分积。当乘数的位是 1 时,部分积与被乘数就是一样的。当乘数的位是 0 时,部分积就是 0。将每一个相继的部分积向左移动一位。

步骤 4: 将相继的部分积和其前面部分积的和相加,从而得到最终的积。

步骤 5: 如果步骤 1 所确定的符号位是负的,就对积取补码。如果是正的,积就保持为原码。

例 2.22 把带符号二进制数相乘:01010011(被乘数)和 11000101(乘数)。

解:

步骤 1: 被乘数的符号位是 0,而乘数的符号位是 1。积的符号位将是 1(负数)。

步骤 2: 取乘数的补码,把它变为原码:

$$11000101 \rightarrow 00111011$$

步骤 3 和步骤 4: 相乘过程如下所示。注意在这些步骤中只使用了数值位。

1010011	被乘数
× 0111011	乘数
1010011	第一个部分积
+ 1010011	第二个部分积
11111001	第一和第二个部分积的和
+ 0000000	第三个部分积
011111001	和
+ 1010011	第四个部分积
1110010001	和
+ 1010011	第五个部分积
100011000001	和
+ 1010011	第六个部分积
1001100100001	和
+ 0000000	第七个部分积
1001100100001	最后的积

步骤 5: 由于积的符号是 1,正如步骤 1 中所确定的那样,因此求积的补码。

$$1001100100001 \longrightarrow 0110011011111$$

粘贴符号位 → **1 0110011011111**

相关问题: 通过转变成十进制数,并完成乘法运算,证明以上相乘过程是正确的。

2.7.4 除法

除法中的数分别是被除数、除数及商,下面的这个标准除法格式对其进行了说明:

$$\frac{被除数}{除数} = 商$$

计算机中的除法运算是通过减法完成的。由于减法是由加法器来完成的,所以除法也可以通过加法器来完成。

除法的结果称为商,即被除数中可以被减去多少次除数的次数等于商,如下面的 21 除以 7 所示的一样:

21	被除数
− 7	第一次减去除数
14	第一个部分余数
− 7	第二次减去除数
7	第二个部分余数
− 7	第三次减去除数
0	余数为 0

在这个简单的例子中，在余数为 0 之前除数从被除数中减去了三次。所以商是 3。

商的符号取决于被除数和除数的符号，依据以下两条规则：

- **如果符号相同，商就是正的。**
- **如果符号不同，商就是负的。**

当两个二进制数相除时，这两个数都必须是原码(非补码)形式。除法过程的基本过程是

步骤 1：确定被除数和除数的符号是相同还是不同。这将确定商的符号。商的初始值是 0。

步骤 2：使用补码加法把除数从被除数中减去，得到第一个部分余数，同时将商加 1。如果这个部分余数是正的，转到步骤 3。如果部分余数是 0 或者是负的，除法就完成了。

步骤 3：从部分余数中减去除数，商加上 1。如果结果是正的，重复以上步骤得到下一个部分余数。如果结果是零或负的，完成除法。

继续从被除数和部分余数中减去除数，直至出现 0 或者负数结果。计算除数被减的次数，就会得到商。例 2.23 使用 8 位带符号二进制数给出了这些步骤。

例 2.23 01100100 除以 00011001。

解：

步骤 1：这两个数的符号都是正的，所以商也是正的。商的初值为零：0000000。

步骤 2：使用补码加法从被除数中减去除数(记住舍去最后的进位)：

01100100	被除数
+ 11100111	除数的补码
01001011	第一个正的部分余数

商加 1：0000000 + 00000001 = 00000001.

步骤 3：使用补码加法从第一个部分余数中减去除数：

01001011	第一个部分余数
+ 11100111	除数的补码
00110010	第二个正的部分余数

步骤 4：使用补码加法从第二个部分余数中减去除数：

00110010	第二个部分余数
+ 11100111	除数的补码
00011001	第三个正的部分余数

商加 1：00000010 + 00000001 = 00000011.

步骤 5：使用补码加法从第三个部分余数中减去除数：

00011001	第三个部分余数
+ 11100111	除数的补码
00000000	余数 0

商加 1:00000011 + 00000001 = **00000100**(最终的商),计算过程就完成了。

相关问题: 转换为十进制数并完成该除法运算,证明以上过程是正确的。

2.8 十六进制数

十六进制计数系统具有 16 个数位,主要作为一种显示或者书写二进制数的简洁方法,因为二进制数和十六进制数之间非常容易转变。正如读者已经注意到的那样,很长的二进制数阅读和书写都很困难,因为很容易丢失或者颠倒某个位。由于计算机和微处理器只能识别 1 和 0,在“机器语言”的编程中,就需要用到 0 和 1。想象一下,为一个以 0 和 1 为基础的微处理器系统编写一个十六位指令时使用十六进制或者八进制将更加有效:八进制将在 2.9 节中介绍。十六进制广泛应用于计算机和微处理器应用程序中。

学完本节以后,应当能够

- 列出十六进制数位(数和字母)
- 以十六进制计数
- 把二进制转换为十六进制
- 把十六进制转换到二进制
- 把十六进制转换到十进制
- 把十进制转换到十六进制
- 进行十六进制数加法运算
- 确定十六进制数的补码
- 进行十六进制数减法运算

◇ 十六进制计数系统由数字 0~9 和字母 A~F 组成

十六进制计数系统的基是 16,也就是,它由 16 个数字和字母共同组成。大多数计数系统都能成组处理二进制数,即多个 4 位的组合,这就使得十六进制非常方便,因为每一个十六进制数就代表了一个 4 位二进制数(如表 2.3 所示)。

表 2.3 十进制数、二进制数、十六进制数的对应

十进制数	二进制数	十六进制数
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9

(续表)

十进制数	二进制数	十六进制数
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

10 个数字数位和 6 个字母字符构成了十六进制计数系统。使用字母 A、B、C、D、E、F 表示数字，初看起来很奇怪，但是请记住任何一种计数系统都是有序符号的集合而已。如果理解了这些符号表示哪些数，只要熟练使用就会十分方便。我们将使用下标 16 来指明十六进制数，以避免和十进制数相混淆。有时，可能会看到十六进制数后面会跟随一个" h"。

2.8.1　十六进制计数

计到 F 时，怎样在十六进制中计数呢？只要到左侧一列继续计数就可以了，如下所示：

10,11,12,13,14,15,16,17,18,19,1A,1B,1C,1D,1E,1F,20,21,22,23,24,
25,26,27,28,29,2A,2B,2C,2D,2E,2F,30,31,...

利用两个十六进制位，你可以数到 FF_{16}，也就是十进制数中的 255。当大于 255 时，就需要三个十六进制数位。比如，100_{16} 在十进制数中就是 256，101_{16} 在十进制数中就是 257，以此类推。最大的 3 位十六进制数是 FFF_{16}，或者十进制数 4095。最大的 4 位十六进制数是 $FFFF_{16}$，也就是十进制数 65 535。



计算机小知识

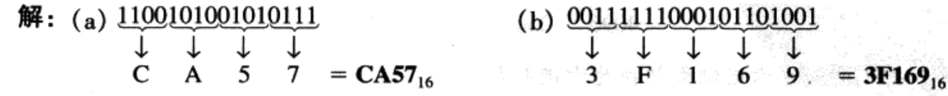
对于千兆字节计的计算机存储器，以二进制指定存储地址是十分麻烦的。例如，在一个 4 GB 的存储器中，指定一个地址就得需要 32 个数位。而使用 8 个十六进制数位来表示 32 位编码就要方便多了。

2.8.2　二进制到十六进制的转换

把二进制数转换为十六进制数是非常直接的过程。将二进制数每 4 位分成一组，从最右边一位开始，用对等的十六进制符号替代相应的每个 4 位数组。

例 2.24　把下面的二进制数转换为十六进制数：

(a)1100101001010111　　　(b)111111000101101001



在(b)部分中添加了两个零，以完成左边的 4 位数组。

相关问题：将二进制数 1001111011110011100 转换为十六进制数。

2.8.3 十六进制到二进制的转换

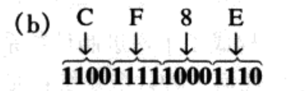
◇ 十六进制和二进制之间的转换直接而且容易。

把十六进制数转换为二进制数的过程,和上述过程相反,用对等的4位数组替代每个十六进制符号,如例2.25所示。

例2.25 确定下面的十六进制数的二进制数。

(a)$10A4_{16}$ (b)$CF8E_{16}$ (c)9742_{16}

解: (a) 1 0 A 4 → 1000010100100 (b) C F 8 E → 1100111110001110 (c) 9 7 4 2 → 1001011101000010



在(a)部分中,认为MSB的前面还有三个0,因而构成了一个4数位数组。

相关问题: 将十六进制数6BD3转换为二进制数。

应该清楚,处理十六进制数要比处理相应的二进制数要容易一些。因为它们之间的转换很容易,所以十六进制系统在程序设计、打印输出及显示中得到了广泛的应用。

2.8.4 十六进制到十进制的转换

一种求十六进制相对应的十进制数的方法是,首先把十六进制数转换为二进制数,然后再把二进制数转换为十进制数。

例2.26 把下面的十六进制数转换为十进制数:

(a)$1C_{16}$ (b)$A85_{16}$

解: 记住,首先将十六进制数转换为二进制数,然后转换为十进制数。

(a) 1 C → 0001 1100

$$00011100 = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28_{10}}$$

(b) A 8 5 → 1010 1000 0101

$$101010000101 = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693_{10}}$$

相关问题: 将十六进制数6BD转变为十进制数。

另一种把十六进制数转换为十进制数的方法是,把每一个十六进制数位的十进制值都乘以这位的权,然后再把这些积加起来。十六进制数的权是16的递增幂(从右到左)。对于4位十六进制数,它的权是

16^3	16^2	16^1	16^0
4096	256	16	1

例2.27 把下面的十六进制数转换为十进制数:

(a)$E5_{16}$ (b)$B2F8_{16}$

解: 回忆表2.3,其中字母A到F分别表示十进制数10到15。

(a) $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229_{10}}$

(b) $B2F8_{16} = (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1)$
$= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1)$
$= 45\,056 + 512 + 240 + 8 = \mathbf{45\,816_{10}}$

相关问题：把 $60A_{16}$ 转换为十进制数。

2.8.5 十进制到十六进制的转换

用 16 重复除十进制数，就会生成相应的十六进制数，由相除的余数所构成。生成的第一个余数是最低有效位。以 16 作为除数的每一次相除都会生成一个余数，这个余数就是相应十六进制数的一个数位。这个过程和 2.3 节提到的十进制到二进制转换中重复除以 2 的方法相似。例 2.28 展示了这个过程。注意当商有小数部分时，用除数乘小数部分就会得到余数。

例 2.28 用重复除以 16 的方法把十进制数 650 转换为十六进制数。

解：

十六进制余数

$\frac{650}{16} = 40.625 \rightarrow 0.625 \times 16 = 10 = A$

(十六进制数)

$\frac{40}{16} = 2.5 \rightarrow 0.5 \times 16 = 8 = 8$

$\frac{2}{16} = 0.125 \rightarrow 0.125 \times 16 = 2 = 2$

当整数的商为零时停止

2 8 A 十六制数

MSD LSD

相关问题：把十进制数 2591 转变为十六进制数。

2.8.6 十六进制加法

◇ 一个计算器可以对十六进制数进行数学运算。

加法可以直接使用十六进制数来进行，记住十六进制数字 0 到 9 等同于十进制数字 0 到 9，而十六进制数字 A 到 F 等同于十进制数 10 到 15。当两个十六进制数相加时，使用下面的方法。（十进制数由下标 10 表示。）

1. 在加法问题任一给定的列中，把两个十六进制数字看成它们的十进制值。例如，$5_{16} = 5_{10}$ 和 $C_{16} = 12_{10}$。
2. 如果这两个数字的和是 15_{10} 或者小些，记下相应的十六进制数字。
3. 如果这两个数字的和大于 15_{10}，记下超出 16_{10} 的量，并在下一列进 1。

例 2.29 把下面的十六进制数相加：

(a) $23_{16} + 16_{16}$ (b) $58_{16} + 22_{16}$ (c) $2B_{16} + 84_{16}$ (d) $DF_{16} + AC_{16}$

解：

(a)
23_{16}
$+16_{16}$
$\mathbf{39_{16}}$

右栏：$3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16}$
左栏：$2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16}$

(b) 58_{16} 右栏： $8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16}$
$+ 22_{16}$ 左栏： $5_{16} + 2_{16} = 5_{10} + 2_{10} = 7_{10} = 7_{16}$
$\mathbf{7A_{16}}$

(c) $2B_{16}$ 右栏： $B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16}$
$+ 84_{16}$ 左栏： $2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16}$
$\mathbf{AF_{16}}$

(d) DF_{16} 右栏： $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$
$+ AC_{16}$ $27_{10} - 16_{10} = 11_{10} = B_{16}$ **加1进位**
$\mathbf{18B_{16}}$ 左栏： $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$
$24_{10} - 16_{10} = 8_{10} = 8_{16}$ **加1进位**

相关问题:$4C_{16}$加上 $3A_{16}$。

2.8.7 十六进制减法

如同前面所学的,通过补码可以实现利用加二进制数来进行减的运算。由于十六进制数可以表示二进制数,所以也可以表示二进制数的补码。

有三种方法可以取得十六进制数的补码。方法 1 最常见也最容易使用。方法 2 和方法 3 是替代方法。

方法 1 将十六进制数转换为二进制数。取二进制数的补码。把结果转换为十六进制数,如图 2.4 所示。

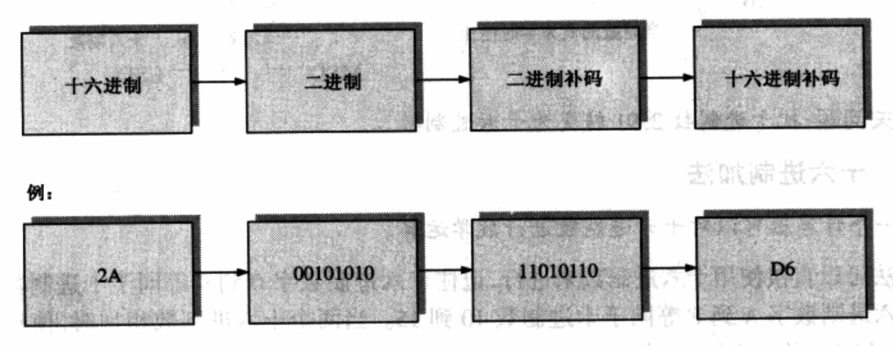


图 2.4 得到十六进制数的补码,方法 1

方法 2 从最大十六进制数减去当前十六进制数,并加 1,如图 2.5 所示。

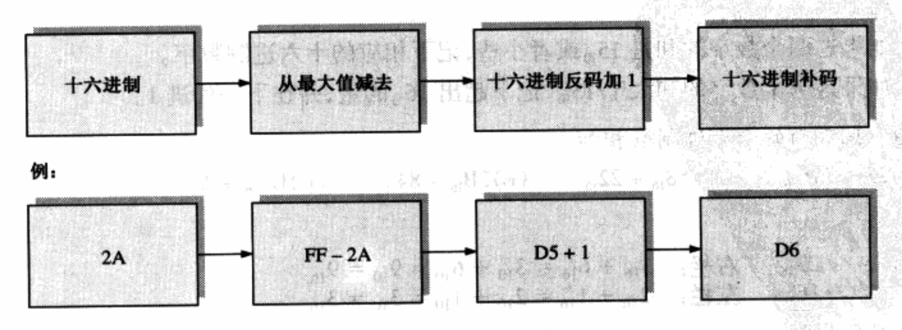


图 2.5 得到十六进制数的补码,方法 2

方法 3 写出单十六进制数字的序列。在这一行的下面，以相反的顺序写出这个序。每个十六进制数字的反码就是该数字正下方的数字。结果数加 1 得到补码，如图 2.6 所示。

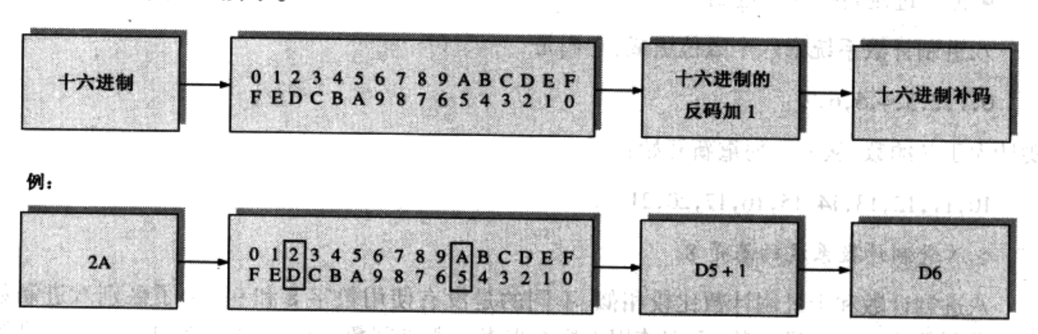


图 2.6 得到十六进制数的补码，方法 3

例 2.30 对下面的十六进制数进行减法计算：

(a) $84_{16} - 2A_{16}$ (b) $C3_{16} - 0B_{16}$

解：

(a) $2A_{16} = 00101010$

$2A_{16}$ 的补码 $= 11010110 = D6_{16}$（使用方法 1）

$$\begin{array}{r} 84_{16} \\ +\ D6_{16} \\ \hline \not{1}5A_{16} \end{array}$$

加

丢弃进位，像补码加法一样

差是 $\mathbf{5A_{16}}$。

(b) $0B_{16} = 00001011$

$0B_{16}$ 的补码 $= 11110101 = F5_{16}$（使用方法 1）

$$\begin{array}{r} C3_{16} \\ +\ F5_{16} \\ \hline \not{1}B8_{16} \end{array}$$

加

丢弃进位

差是 $\mathbf{B8_{16}}$。

相关问题： 从 BCD_{16} 中减去 173_{16}。

2.9 八进制数

与十六进制计数系统相似，八进制计数系统提供了一种简洁的方法来表示二进制数和编码。然而，在计算机和微处理器的输入和输出时，需要表示二进制的量，这时十六进制使用得更频繁。

学完本节以后，应当能够

- 写出八进制计数系统的数字
- 把八进制转换为十进制

- 把十进制转换为八进制
- 把八进制转换为二进制
- 把二进制转换为八进制

八进制计数系统由八个数位组成,它们是

0,1,2,3,4,5,6,7

要计大于7的数,从另一列重新开始:

10,11,12,13,14,15,16,17,20,21...

◇ 八进制计数系统的基是8。

八进制计数和十进制计数比较相似,不同的是没有使用数字8和9。为了区别八进制数和十进制数或者十六进制数,我们使用下标8来表示八进制数。比如,八进制 15_8 等于十进制数 13_{10} 和十六进制数 D。有时会看到八进制数后面跟随"o"或者"Q"。

2.9.1 八进制数到十进制数的转换

由于八进制数的基是8,因此每一个相继的数位都是8的递增幂,从最右边 8^0 开始,计算八进制数的相应十进制数数值,可以通过把每一个数字都乘以其相应的权,并把所得到的积加起来,下面展示了 2374_8 的转换过程。

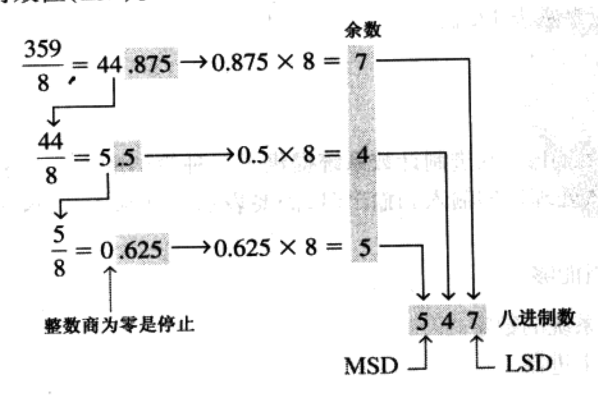
权: $8^3\ 8^2\ 8^1\ 8^0$

八进制数: 2 3 7 4

$$\begin{aligned} 2374_8 &= (2\times 8^3) + (3\times 8^2) + (7\times 8^1) + (4\times 8^0) \\ &= (2\times 512) + (3\times 64) + (7\times 8) + (4\times 1) \\ &= 1024 + 192 + 56 + 4 = 1276_{10} \end{aligned}$$

2.9.2 十进制数到八进制数的转换

把十进制数转换为八进制数的方法是,重复除以8的方法,这和十进制到二进制或十六进制的转换方法相似。为了给出这个过程,我们把十进制数359转换为八进制数,以8为除数的每一次相继相除都会产生一个余数,而这个余数就会成为相应八进制数的数位。所产生的第一个余数是最低有效位(LSD)。

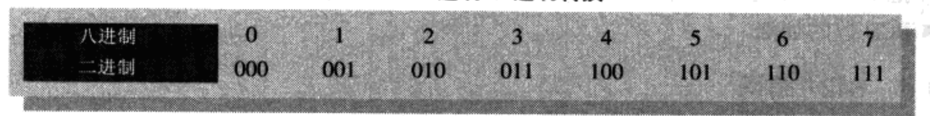


2.9.3　八进制到二进制的转换

◇ 八进制是表示二进制数的简洁方法，但是并没有十六进制常用。

因为每一个八进制数位都可以由一个 3 位二进制数来表示，所以八进制转换为二进制数很容易。每一个八进制数位由 3 位二进制数表示，如表 2.4 所示。

表 2.4　八进制/二进制转换



八进制	0	1	2	3	4	5	6	7
二进制	000	001	010	011	100	101	110	111

为了把八进制数转换为二进制数，只要用合适的 3 位二进制数替换每个八进制数字就可以了。这个过程如例 2.31 所示。

例 2.31　把下面的八进制数转换为二进制数：

(a)13_8　(b)25_8　(c)140_8　(d)7526_8

解：

(a) 1 3 → 001011　(b) 2 5 → 010101　(c) 1 4 0 → 001100000　(d) 7 5 2 6 → 111101010110

相关问题：将每一个二进制数转变为十进制数，并证实每一个值和相应八进制数的十进制值相等。

2.9.4　二进制到八进制的转换

二进制数到八进制数的转换是八进制数到二进制数转变的逆过程。如下所示。从最右边的 3 位数一组开始，从右向左移动，把每 3 位数一组转变为相应的八进制数。如果最左边的一组没有可用的 3 位数，就加上一个或者两个 0 以形成完整的一组。前面的 0 不影响二进制数的值。

例 2.32　把下面每一个二进制数都转变为八进制数：

(a)110101　(b)101111001　(c)100110011010　(d)11010000100

解：

(a) 110101 → 6 5 = $\mathbf{65_8}$

(b) 101111001 → 5 7 1 = $\mathbf{571_8}$

(c) 100110011010 → 4 6 3 2 = $\mathbf{4632_8}$

(d) 011010000100 → 3 2 0 4 = $\mathbf{3204_8}$

相关问题：把进制数 1010101000111110010 转换为八进制数。

2.10　二－十进制码(BCD)

二－十进制码(BCD)是一种以二进制编码表示每一个十进制数字的方法。在 BCD 系统

中,只有 10 个编码组,所以很容易在十进制和 BCD 之间进行转换。因为要以十进制读写,BCD 编码提供了二进制系统的一个良好接口。这种接口的例子是键盘输入和数字读出。

学完本节以后,应当能够

- 把每一个十进制数字转变为 BCD
- 以 BCD 表示十进制数
- 把 BCD 转变为十进制
- BCD 数的加法

2.10.1 8421 编码

◇ 在 BCD 码中,每个十进制数都由 4 位二进制编码表示。

8421 编码是 BCD(二 - 十进制码)编码的一种类型。二 - 十进制意思是,每一个十进制数(从 0 到 9)都由 4 位二进制编码表示。名称 8421 表明了 4 个位的二进制权($2^3,2^2,2^1,2^0$)。这种编码的优点是,8421 编码数和我们熟悉的十进制数之间很容易转变。只要记住 10 个十进制数字的二进制组合,如表 2.5 所示。8421 编码是主要的 BCD 编码并且当我们提及 BCD 时,总是指 8421 编码,除非有特殊的说明。

表 2.5 十进制/BCD 转变

十进制	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

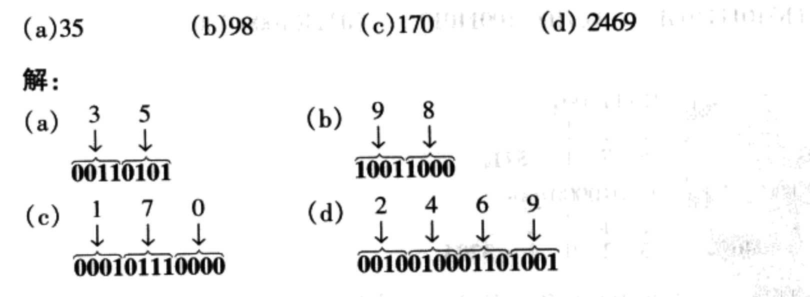
无效码 应当认识到,使用 4 个位可以表示 16 个数(从 0000 到 1111),但是在 8421 码中,这 16 个数当中只使用了 10 个数。未使用的编码组合为 1010,1011,1100,1101,1110,1111。它们在 8421 BCD 码中是无效的。

为了以 BCD 码表示任意十进制数,只要将每个十进制数位用合适的 4 位编码替代就可以了,如例 2.33 所示。

例 2.33 把下面的十进制数转换为 BCD:

(a)35 (b)98 (c)170 (d) 2469

解:

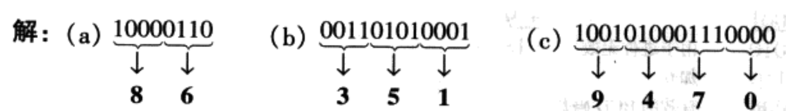


相关问题:把十进制数 9673 转换为 BCD 码。

根据一个 BCD 数确定一个十进制数,同样很容易。从最右边的一位开始,把 BCD 码分成四位一组。然后写出每个 4 位一组所表示的十进制数字。例 2.34 给出了这个过程。

例 2.34 把下面的 BCD 编码转换为十进制数:

(a)10000110　　(b)001101010001　　(c)1001010001110000



相关问题：把 BCD 编码 10000010001001110110 转换为十进制数。

2.10.2　BCD 码加法

BCD 是一种数字码，并且可以使用在算术运算中。加法是最重要的运算，因为其他三种运算（减法、乘法、除法）可以用加法来完成。下面介绍了如何相加两个 BCD 码数：

步骤 1：使用 2.4 节中二进制加法的法则，相加两个 BCD 数。

步骤 2：如果 4 位和等于或者小于 9，这个和就是一个合法 BCD 码数。

步骤 3：如果 4 位和大于 9，或者如果在 4 位一组之外产生了一个进位，那么这就是一个无效结果。在 4 位和上加上 6(0110) 以跳过 6 个无效状态，并将编码返回 8421。如果加上 6 时产生进位，就把这个进位加到下一个 4 位一组中。

例 2.35 说明了 BCD 码的加法，在这个例子中，4 位和等于或者小于 0，所以这些 4 位和都是有效的 BCD 码数。例 2.36 说明了具有无效和（大于 9 或者有进位）的过程。

例 2.35　将下面的 BCD 码数相加：

(a)0011 + 0100　　(b)00100011 + 00010101

(c)10000110 + 00010011　　(d)010001010000 + 010000010111

解：下面也给出了十进制加法以做对比。

(a)
```
  0011      3
+ 0100    + 4
  0111      7
```

(b)
```
  0010 0011     23
+ 0001 0101   + 15
  0011 1000     38
```

(c)
```
  1000 0110     86
+ 0001 0011   + 13
  1001 1001     99
```

(d)
```
  0100 0101 0000     450
+ 0100 0001 0111   + 417
  1000 0110 0111     867
```

注意在每一种情况下，任一 4 位列的和都没有超出 9，所以这些结果都是有效的 BCD 码数。

相关问题：将 BCD 码数相加：1001000001000011 + 0000100100100101。

例 2.36　相加下面的 BCD 码数：

(a) 1001 + 0100　　(b) 1001 + 1001

(c) 00010110 + 00010101　　(d) 01100111 + 01010011

解：下面也给出了十进制加法以做对比。

(a)
```
        1001                              9
      + 0100                            + 4
        1101    无效的 BCD 码数 (>9)      13
      + 0110    加 6
0001    0011    有效的 BCD 码数
  ↓       ↓
  1       3
```

```
(b)          1001                    9
           + 1001                  + 9
       1     0010    由于进位无效    18
           + 0110    加 6
    0001     1000    有效的 BCD 码数
      ↓        ↓
      1        8

(c)   0001     0110                          16
    + 0001     0101                        + 15
      0010     1011    右边一组无效（>9）     31
                        左边一组有效
             + 0110    加 6 变为有效码，
                        加进位，0001 到下一组
      0011     0001    有效的 BCD 码数
        ↓        ↓
        3        1

(d)          0110        0111                          67
           + 0101        0011                        + 53
             1011        1010    两组都无效（>9）     120
           + 0110      + 0110    加 6 到两组
    0001     0010        0000    有效 BCD 码
      ↓        ↓           ↓
      1        2           0
```

相关问题：将 BCD 码数相加：01001000 + 00110100。

2.11 数字编码

在计数系统中，有许多专用编码。我们刚刚学习了 BCD 编码，现在就看看其他的一些编码。有些编码是严格的数字，比如 BCD，而其他的则是字母数字；也就是说，它们用以表示数字、字母、符号及指令。本节所要介绍的编码是格雷(Gray)码和 ASCII 编码。

学完本节之后，你应当能够

- 解释格雷码的优点
- 在格雷码和二进制之间相互转换
- 使用 ASCII 编码

2.11.1 格雷码

格雷码是无权码，也不是算术编码；也就是说，格雷码没有赋予位单元特定的权。格雷码的重要特征是，从一个编码字到下一个接续编码字仅有一位发生了变化。这个特征在许多应用程序中是非常重要的，比如轴位编码器，其中在两个相邻顺序数之间，错误敏感度随着位数改变数目的增加而增加。

◇ 格雷码的一位改变的特征减小了出错概率。

表 2.6 列出了十进制数 0 到 15 所对应的 4 位格雷码。表中给出了二进制以做参照。和二进制数相似，格雷码可以拥有任意的位数。注意两个相邻的格雷码字之间的一位变化。比如，从十进制数 3 到十进制数 4，格雷码从 0010 变为 0110，而二进制编码从 0011 变为 0100，改变了 3 个位。格雷码中唯一的位改变，就是从右数的第 3 位，其他位保持不变。

表 2.6　4 位格雷码

十进制	二进制	格雷码	十进制	二进制	格雷码
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

二进制到格雷码的转换　二进制和格雷码之间的转换有时很有用。下面的规则解释了怎样把二进制数转换为格雷码字：

1. 格雷码中的最高有效位(最左边)等同于二进制数中相应的最高有效位(MSB)；
2. 从左到右，加上每一对相邻的二进制编码位，从而得到下一个格雷码位。舍去进位。

例如，二进制数 10110 到格雷码的转换如下：

$$
\begin{array}{ccccccccccc}
1 & - + \rightarrow & 0 & - + \rightarrow & 1 & - + \rightarrow & 1 & - + \rightarrow & 0 & & \text{二进制} \\
\downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \\
1 & & 1 & & 1 & & 0 & & 1 & & \text{格雷码}
\end{array}
$$

格雷码是 11101。

格雷码到二进制的转换　为了把格雷码转换为二进制数，使用相似的方法，但是有一些区别。应用下面的规则：

1. 二进制编码中的最高有效位(最左边)等于格雷码中相应的位；
2. 将所产生的每个二进制编码位加上下一相邻位置的格雷码位。舍去进位。

例如，格雷码 11011 到二进制的转换如下：

$$
\begin{array}{ccccccccccc}
1 & & 1 & & 0 & & 1 & & 1 & & \text{格雷码} \\
\downarrow & \nearrow^{+} & \downarrow & \nearrow^{+} & \downarrow & \nearrow^{+} & \downarrow & \nearrow^{+} & \downarrow & & \\
1 & & 0 & & 0 & & 1 & & 0 & & \text{二进制}
\end{array}
$$

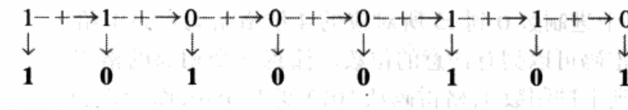
二进制数是 10010。

例 2.37　(a)把二进制数 11000110 转换为格雷码。

(b)把格雷码 10101111 转换为二进制数。

解：

(a)二进制数到格雷码：



(b)格雷码到二进制数：

1 0 1 0 1 1 1 1

1 1 0 0 1 0 1 0

相关问题：(a)把二进制数 101101 转换为格雷码。(b)把格雷码 100111 转换为二进制数。

2.11.2 应用举例

3 位轴位编码器机械构造的简化图如图 2.7 所示，其中有三个同心导电圆圈被分成 8 个扇区。扇区越多，位置就能表示得越准确，但是为了解释的方便，仅仅使用了 8 个色扇。每个圆圈的每个扇区处于高电位或处于低电位，分别表示 1 和 0。1 由有色扇区表示，而 0 由白色扇区所表示。当圆圈围绕轴旋转时，就会和一个电刷装置发生接触，这个电刷放置在一个固定的位置，并连接输出线路。当轴位逆时针旋转 360°时，8 个扇区就会经过三个电刷，产生一个 3 位二进制输出，这个输出指示了轴位的位置。

在图 2.7(a)中，这些扇区以直接二进制样式排列，所以电刷从 000 到 001，再到 010，等等。当电刷位于有色扇区时，输出为 1；在白色扇区时，输出为 0。如果一个电刷从一个扇区到下一个扇区的转移瞬间中，仅仅稍微先于其他的电刷，就可能发生错误输出。考虑一下，当电刷位于 111 扇区、准备进入 000 扇区时会发生什么。如果最高位 MSB 电刷略微在前，这个位置就会被转移瞬间的 011 错误地显示，而不是 111 或者 000。在这种应用类型中，保持所有电刷机械上的精确对齐事实上是不可能的；所以，扇区的转移瞬间中总是会发生一些错误。

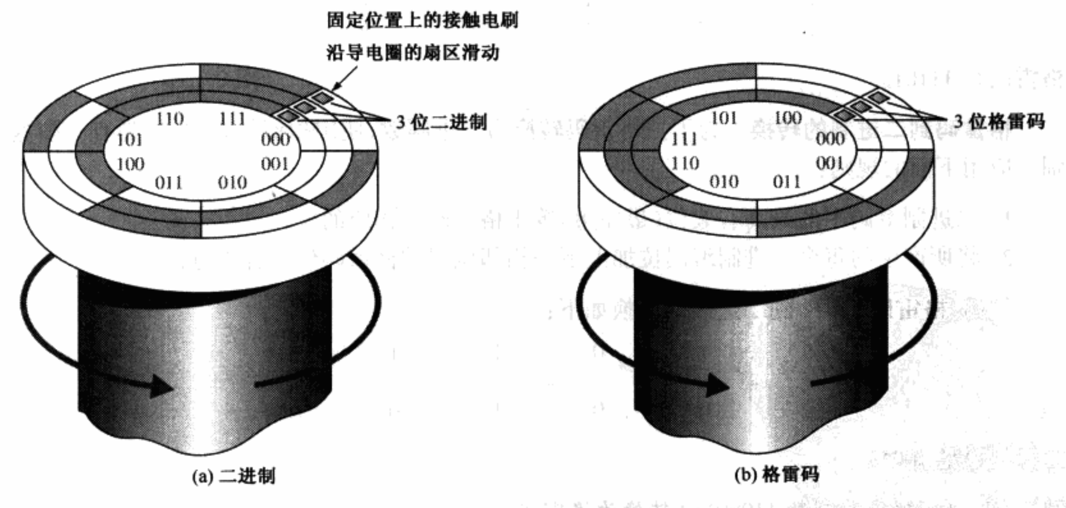


图 2.7 一个简单的例子，解释在轴位编码器中格雷码是如何解决出错问题的

格雷码用来消除二进制所固有的这类出错问题。如图 2.7(b)所示，格雷码保证了在两个相邻的扇区中，只有一个位会发生改变。这就意味着即使电刷没有精确对准，也绝对不会发生

转移瞬间中的错误。例如,让我们再次考虑一下,当电刷位于 111 扇区、准备进入下一扇区 101 时会发生什么。无论电刷是否对准,在转移瞬间只有两个可能的输出 111 和 101。在其他扇区的转移瞬间中,也会产生相同的结果。

2.12　错误检测和校验码

在这一节,将讨论在数码中添加位的两种方法,用以检测一位错误或纠正一位错误。我们介绍错误检测的奇偶校验,以及一个有关错误检测和纠正的汉明(Hamming)码方法。当发现一个给出的码字的一位有错误时,可以简单地对这位取反来纠正它。

学完本节以后,应当能够

- 使用奇偶校验位来判断数码中是否有错误
- 在一个数码中分配一个恰当的奇偶校验位
- 使用汉明码检测一个错误和纠正它
- 对一个错误的纠正分配恰当的奇偶校验位

◇ 通过奇偶校验位可以得知 1 的个数是奇数还是偶数。

2.12.1　错误检测的奇偶校验方法

许多系统都使用一个奇偶校验位作为位错误检测的手段。任意的多位数组都包含奇数个 1 或偶数个 1。一个奇偶校验位附加到多位数组中,使得这组数中 1 的个数总是偶数或者总是奇数。一个偶校验位使得 1 的总数为偶数,而奇校验位使得 1 的总数为奇数。

一个给定的系统运行于偶校验或者奇校验,而不是同时作用于两者。例如,如果某系统运行于偶校验,对于所接收的每一个多位数组都做一个检查,以确保这个多位数组中 1 的总数是偶数。如果有奇数个 1,就有一个错误发生了。

作为对奇偶校验位怎样附加到数编码中的一个说明,表 2.7 列出了每个 BCD 数的偶校验和奇校验的奇偶校验位。

表 2.7　带奇偶校验位的 BCD 码

偶校验		奇校验	
校验位 P	BCD	校验位 P	BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

奇偶校验位可以附加到数码的开头或者结尾,这取决于系统的设计。注意,1 的总数,包括奇偶校位上的 1,对于偶校验总是偶数,对于奇校验总是奇数。

检测一个错误 奇偶校验位提供了单个位错误的检测(或者任何奇数个错误,这种可能性较小),但是不能检测一组中的两个错误。比如,假设我们希望传送 BCD 编码 0101。(奇校验可以应用于任何位数;使用 4 位作为说明。)所传送的总的数码,包括偶校验位如下所示:

00101 (偶校验位;BCD 码)

现在,让我们假设从左边数第 3 位发生了错误(1 变为 0),如下所示:

00001 (偶校验位;一位错误)

当该数码被接收时,奇偶校验检测电路检测出只有一个 1(奇数),而应当有偶数个 1。因为在数码被接收时,偶数个 1 没有出现在数码中,所以就指明了一个错误。

一个奇校验位提供了相似的方式,用以检测给定多位数组的单个错误。

例 2.38 给下面的数码组加上适当的偶校验位:

(a)1010 (b)111000 (c)101101 (d)1000111001001 (e)101101011111

解:根据需要使奇偶校验位为 1 或者 0,使得 1 的总数为偶数。奇偶校验位将是最左边的一位。

(a)01010 (b)1111000 (c) 0101101 (d)0100011100101 (e)1101101011111

相关问题:为字母 K 的 7 位 ASCII 码加上一个偶校验位。

例 2.39 一个奇校验系统接收到下面的数码组:10110、11010、110011、110101110100 和 1100010101010。如果有错误,确定哪个组是错误的。

解:由于需要奇校验,所以任何一个拥有偶数个 1 的小组就是错误的。下面的组有错误:**110011 和 1100010101010**。

相关问题:一个奇校验系统接收到下列 ASCII 字符:00110111。这个数组正确吗?

2.12.2 汉明纠错码

如同所见,一个奇偶校验位可以检测数码字中的一个位错误。一个奇偶校验位可以指出在某个多位数组中有一个错误。为了纠正一个检测到的错误,就需要更多的信息,因为在纠错前必须判断错误位所在的位置。为了能够纠正检测到的错误,就需要多于一个的校验位。在 7 位码中,有 7 个数位可能发生错误。在这种情况下,三个校验码不仅可以检查错误,而且可以判断出错位的位置。汉明码提供了一位错误的纠正方法。下面的讲解给出了纠正一位错误的 7 位汉明码的组成。

奇偶校验位的数目 如果定义数据的总位数为 d,校验位的位数为 p,p 由下面的关系确定:

$$2^p \geq d + p + 1 \tag{2.1}$$

例如,如果有四位数据位,然后 p 由式(2.1)尝试来确定是否满足条件。让 $p = 2$,得到

$$2^p = 2^2 = 4 \quad 和 \quad d + p + 1 = 4 + 2 + 1 = 7$$

因为 2^p 必须等于或大于 $d + p + 1$,所以式(2.1)没有满足,必须再尝试一次,让 $p = 3$,然后

$$2^p = 2^3 = 8 \quad 和 \quad d + p + 1 = 4 + 3 + 1 = 8$$

p 的值满足式(2.1),因此需要 3 位校验位来检测 4 位数据的数位错误。需要注意的是,检错和纠错是对所有的数位,即校验位和数据位,在数码中校验位也进行自身校验。

数码中奇偶校验位的位置　在以上的具体例子中,已经求得了奇偶校验位的数目,然后必须把它们放到数码中的恰当位置。在这一点,应该认识到例子中数码由四个数据位和三个校验位组成。最左位定义为位 1,次位为位 2,以此类推:

位 1,位 2, 位 3, 位 4,位 5,位 6,位 7

校验位所处的位置对应于递增的 2 的幂次。如下给出:

$$P_1, \quad P_2, \quad D_1, \quad P_3, \quad D_2, \quad D_3, \quad D_4$$

符号 P_n 定义为特定的校验位,D_n 定义为特定的数据位。

校验位值的确定　最后,必须给每个校验位赋予恰当的值 0 或 1。因为每个校验位提供了对整个数码中其他位的校验,所以必须知道其他位的值以便给校验赋值。为了确定位的值,首先给二进制数中的每个位的位置赋值,即写出每个十进制位置数和所对应的二进制位置数,如表 2.8 的第二行和第三行给出。下一步指出校验位和数据位所处的位置,如表 2.8 的第一行所示。注意校验位 P_1 的二进制位置数的最右位为 1。此校验位检测所有的位位置,包括自身,所对应的二进制位置数中相同位置(最右位)的 1。因此,校验位 P_1 检测位位置 1、3、5 和 7。

表 2.8　7 位纠错码位位置表

位名称	P_1	P_2	D_1	P_3	D_2	D_3	D_4
位位置	1	2	3	4	5	6	7
二进制位置数	001	010	011	100	101	110	111
数据位 (D_n)							
校验位 (P_n)							

校验位 P_2 的二进制位置数的中间位为 1,它检测所有的位位置,包括自身,所对应的二进制位置数中相同位置(中间位)的 1。因此,校验位 P_2 检测位位置 2、3、6 和 7。

校验位 P_3 的二进制位置数的最左位为 1,它检测所有的位位置,包括自身,所对应的二进制位置数中相同位置(最左位)的 1。因此,校验位 P_3 检测位位置 4、5、6 和 7。

以上每一种情况,校验位要求被赋予 0 或 1,以使得所有被检测数位的 1 的个数为奇数或偶数。下面的例子会详细讲解。

例 2.40　为 BCD 码 1001(数据位)确定汉明码,使用偶校验。

解:

步骤1:确定所需要的校验位的个数。假定 $p=3$,那么

$$2^p = 2^3 = 8$$
$$d + p + 1 = 4 + 3 + 1 = 8$$

三个校验位足够了。

$$码的位数 = 4 + 3 = 7$$

步骤2:建立位位置表,如表2.9所给出,填入数据位,校验位由下面的步骤确定。

表2.9

位名称	P_1	P_2	D_1	P_3	D_2	D_3	D_4
位位置	1	2	3	4	5	6	7
二进制位置数	001	010	011	100	101	110	111
数据位			1		0	0	1
校验位	0	0		1			

步骤3:确定校验位如下:

位 P_1 校验位位置1、3、5和7,因为对应所有被检测数位的1的个数为偶数,所以校验位 P_1 必须是0。

位 P_2 校验位位置2、3、6和7,因为对应所有被检测数位的1的个数为偶数,所以校验位 P_2 必须是0。

位 P_3 校验位位置4、5、6和7,因为对应所有被检测数位的1的个数为偶数,所以校验位 P_2 必须是1。

步骤4:这些校验位填入表2.10,最后结果合成的数码是0011001。

相关问题:为BCD码1000确定汉明码,使用偶校验。

例2.41 为数据位10110确定汉明码,使用奇校验。

解:

步骤1:确定所需要的校验位的个数。这里数据位的位数 d 为5,从前面的例子可知,$p=3$ 不满足条件,尝试 $p=4$:

$$2^p = 2^4 = 16 \text{ 和 } d + p + 1 = 5 + 4 + 1 = 10$$

四个校验位足够了。

$$码的位数 = 5 + 4 = 9$$

步骤2:建立位位置表,如表2.10所给出,填入数据位,校验位由下面的步骤确定。注意 P_4 处在位位置8。

步骤3:确定校验位如下:

位 P_1 校验位位置1、3、5、7和9,因为对应所有被检测数位的1的个数为奇数,所以校验位 P_1 必须是1。

位 P_2 校验位位置 2、3、6 和 7，因为对应所有被检测数位的 1 的个数为奇数，所以校验位 P_2 必须是 0。

位 P_3 校验位位置 4、5、6 和 7，因为对应所有被检测数位的 1 的个数为奇数，所以校验位 P_3 必须是 1。

位 P_4 校验位位置 8 和 9，因为对应所有被检测数位的 1 的个数为奇数，所以校验位 P_4 必须是 1。

步骤 4：这些校验位填入表 2.10，最后结果组成的数码是 101101110。

表 2.10

位名称	P_1	P_2	D_1	P_3	D_2	D_3	D_4	P_4	D_5
位位置	1	2	3	4	5	6	7	8	9
二进制位置数	0001	0010	0011	0100	0101	0110	0111	1000	1001
数据位			1		0	1	1		0
校验位	1	0		1				1	

相关问题：为 11001 确定汉明码，使用奇校验。

2.12.3　使用汉明码检错和纠错

现在已经讲解了如何用汉明码组成一个错误检测码。那么如何去确定一个错误的位置和纠正它呢？每一个校验位和被检测的数位组，必须符合恰当的奇偶校验。如果在一个码字中有三个奇偶校验位，则形成三个奇偶校验。如果在一个码字中有四个奇偶校验位，则形成四个奇偶校验，以此类推。每一个奇偶校验将产生一个正确或错误的结果。所有奇偶校验的结果指出是否有一个位是错误的。如下所示：

步骤 1：从 P_1 所对应的数位组开始；

步骤 2：对这个数位组进行奇偶校验，0 表示奇偶校验正确，1 表示奇偶校验错误；

步骤 3：对每个奇偶校验数组重复步骤 2；

步骤 4：所有奇偶校验的结果形成了一个二进制数，指出了错误的数码位的位置。这就是错误位置码。第一个奇偶校验产生最低位。如果所有检测的结果都正确，那么没有错误。

例 2.42　假设例 2.40 中的码字（0011001）被传输出去，收到的码字是 0010001。接收器不能"鉴别"传输过来的码字，必须求助于恰当的奇偶校验以鉴别这个码字是否正确。如果使用偶校验，指出在传输中发生的错误。

解：首先，建立一个位位置表，如表 2.11 所示。

表 2.11

位名称	P_1	P_2	D_1	P_3	D_2	D_3	D_4
位位置	1	2	3	4	5	6	7
二进制位置数	001	010	011	100	101	110	111
接收到的码	0	0	1	0	0	0	1

第一个奇偶校验：
位 P_1 检测位置 1、3、5 和 7；
数组中有两个 1；
奇偶校验正确 ——→ 0(最低位)
第二个奇偶校验：
位 P_2 检测位置 2、3、6 和 7；
数组中有两个 1；
奇偶校验正确 ——→ 0
第三个奇偶校验：
位 P_3 检测位置 4、5、6 和 7；
数组中有一个 1；
奇偶校验错误 ——→ 1(最高位)

结果：错误位置码是 100(二进制数四)。这表明码字中的第 4 位是错误的。它是 0，但应该是 1。正确的码字应该是 0011001，这和传输出来的码字相符。

相关问题：如果接收到的码是 0111001，重做上面给出的例子。

例 2.43 接收到的码是 101101010。有四个校验位，使用奇校验。

解：首先，建立一个位位置表，如表 2.12 所示。

表 2.12

位名称	P_1	P_2	D_1	P_3	D_2	D_3	D_4	P_4	D_5
位位置	1	2	3	4	5	6	7	8	9
二进制位置数	0001	0010	0011	0100	0101	0110	0111	1000	1001
接收到的码	1	0	1	1	0	1	0	1	0

第一个奇偶校验：
位 P_1 检测位置 1、3、5、7 和 9；
数组中有两个 1；
奇偶校验错误 ——→ 1(最低位)
第二个奇偶校验：
位 P_2 检测位置 2、3、6 和 7；
数组中有两个 1；
奇偶校验错误 ——→ 1
第三个奇偶校验：
位 P_3 检测位置 4、5、6 和 7；
数组中有两个 1；
奇偶校验错误 ——→ 1
第四个奇偶校验：
位 P_4 检测位置 8 和 9；

数组中有一个 1;

奇偶校验正确——————————→0(最高位)

结果:错误位置码是 0111(二进制数七)。这表明码字中的第 7 位是错误的。正确的数码应该是 101101110。

相关问题:如果使用奇校验,接收到的码是 101111001,请纠正错误。

自测题 (答案在本章的结尾)

1. $2\times10^1+8\times10^0$ 等于

   (a)10　(b)280　(c)2.8　(d)28

2. 二进制数 1101 等于下面哪个十进制数:

   (a)13　(b)49　(c)11　(d)3

3. 二进制数 11011101 等于下面哪个十进制数:

   (a)121　(b)221　(c)441　(d)256

4. 十进制数 17 等于下面哪个二进制数:

   (a)10010　(b)11000　(c)10001　(d)01001

5. 十进制数 175 等于下面哪个二进制数

   (a)11001111　(b)10101110　(c)10101111　(d)11101111

6. 11010 + 01111 的和等于

   (a)101001　(b)101010　(c)110101　(d)101000

7. 110 - 010 的差等于

   (a)001　(b)010　(c)101　(d)100

8. 10111001 的反码是

   (a)01000111　(b)01000110　(c)11000110　(d)10101010

9. 11001000 的补码是

   (a)00110111　(b)00110001　(c)01001000　(d)00111000

10. 十进制数 + 122 的补码表示形式为

    (a)01111010　(b)11111010　(c)01000101　(d)10000101

11. 十进制数 - 34 的补码表示形式为

    (a)01011110　(b)10100010　(c)11011110　(d)01011101

12. 单精度浮点二进制数总共有

    (a)8 位　(b)16 位　(c)24 位　(d)32 位

13. 在补码形式中,二进制数 10010011 等于下面哪个十进制数:

    (a) - 19　(b) + 109　(c) + 91　(d) - 109

14. 二进制数 1011001110010101 00001 在八进制中可以写为

    (a)5471230_8　(b)5471241_8　(c)2634521_8　(d)23162501_8

15. 二进制数 1000110101000 1101111 在十六进制中可以写为

    (a) $AD467_{16}$　(b)$8C46F_{16}$　(c)$8D46F_{16}$　(d)$AE46F_{16}$

16. $F7A9_{16}$的二进制数为

    (a)1111011110101001　(b)1110111110101001

    (c)1111111010110001　(d)1111011010101001

17. 十进制数 473 的 BCD 码为
(a)111011010 (b)110001110011 (c)010001110011 (d)010011110011
18. 有偶校验错误的码是
(a)1010011 (b)1101000 (c)1001000 (d)1110111

习题

2.1 节 十进制数

1. 下面每一个十进制数中,数字 6 的权各是多少?
(a)1386 (b)54 692 (c)671 920
2. 把下面的每一个十进制数表示为 10 的幂数:
(a)10 (b)100 (c)10 000 (d)1 000 000
3. 求出下面十进制数中每一位数字的值:
(a)471 (b)9356 (c)125 000
4. 使用 4 个十进制数位最大能数到几?

2.2 节 二进制数

5. 把下面每一个二进制数都转换为十进制数:
(a)11 (b)100 (c)111 (d)1000
(e)1001 (f)1100 (g)1011 (h)1111
6. 把下面每一个二进制数都转换为十进制数:
(a)1110 (b)1010 (c)11100 (d)10000
(e)10101 (f)11101 (g)10111 (h)11111
7. 把下面每一个二进制数都转换为十进制数:
(a)110011.11 (b)101010.01 (c)1000001.111 (d)1111000.101
(e)1011100.10101 (f)1110001.0001 (g)1011010.1010 (h)1111111.11111
8. 求下面所列出位数的二进制数字(位)所能表示的最大十进制数分别是多少?
(a)2 (b)3 (c)4 (d)5 (e)6
(f)7 (g)8 (h)9 (i)10 (j)11
9. 表示下面的十进制数分别需要多少位?
(a)17 (b)35 (c)49 (d) 68
(e)81 (f)114 (g)132 (h)205
10. 为下面每一个十进制序列分别生成二进制序列:
(a)0~7 (b)8~15 (c)16~31 (d) 32~63 (e)64~75

2.3 节 十进制数到二进制数的转换

11. 使用权值和的方法把每一个十进制数都转换为二进制数:
(a)10 (b)17 (c)24 (d)48
(e)61 (f)93 (g)125 (h)186
12. 使用权值的和方法把每一个十进制数都转变为二进制数:
(a)0.32 (b)0.246 (c)0.0981
13. 使用重复除以 2 的方法将每一个十进制数都转换为二进制数:
(a)15 (b) 21 (c)28 (d)34
(e)40 (f) 59 (g)65 (h)73

14. 使用重复乘以 2 的方法把每一个十进制小数都转换为二进制数：
(a)0.98　(b)0.347　(c)0.9028

2.4 节　二进制算术

15. 二进制数的加法：
(a)11 + 01　(b)10 + 10　(c)101 + 11
(d)111 + 110　(e)1001 + 101　(f) 1101 + 1011

16. 对下面的二进制数使用直接减法：
(a)11 - 1　(b)101 - 100　(c)110 - 101
(d)1110 - 11　(e)1100 - 1001　(f)11010 - 10111

17. 执行下面的二进制乘法：
(a)11 × 11　(b)100 × 10　(c)1111 × 101
(d)1001 × 110　(e)1101 × 1101　(f)1110 × 1101

18. 对下面的二进制数进行除法运算：
(a)100 ÷ 10　(b)1001 ÷ 11　(c)1100 ÷ 100

2.5 节　二进制数的反码和补码

19. 确定下面每一个二进制数的反码：
(a) 101　(b)110　(c)1010
(d)11010111　(e)1110101　(f)00001

20. 使用任何一种方法，确定下面每一个二进制数的补码：
(a)10　(b)111　(c)1001　(d)1101
(e)11100　(f)10011　(g)10110000　(h)00111101

2.6 节　带符号数

21. 把下面的每一个十进制数都表示为 8 位带符号数值的二进制数：
(a) + 29　(b) - 85　(c) + 100　(d) - 123

22. 把下面的每一个十进制数都以反码形式表示为一个 8 位数：
(a) - 34　(b) + 57　(c) - 99　(d) + 115

23. 把下面的每一个十进制数都以补码形式表示为一个 8 位数：
(a) + 12　(b) - 68　(c) + 101　(d) - 125

24. 以符号数值形式确定每个带符号二进制数的十进制数值：
(a)10011001　(b)01110100　(c)10111111

25. 以反码形式确定每个带符号二进制数的十进制数值：
(a)10011001　(b)01110100　(c)10111111

26. 以补码形式确定每个带符号二进制数的十进制数值：
(a)10011001　(b)01110100　(c) 10111111

27. 以单精度浮点数表示下面每一个带符号二进制数：
(a)0111110000101011　(b)100110000011000

28. 确定下面单精度浮点数的数值：
(a)1 10000001 01001001110001000000000
(b)0 11001100 10000111110100100000000

2.7 节　带符号数的算术运算

29. 把下面每对十进制数都转变为二进制数，并使用补码形式进行相加运算：
(a)33,15　(b)5, - 27　(c) - 46,25　(d) - 110, - 84

30. 以补码形式执行下面每一个加法：

(a)00010110 + 00110011　　(b)01110000 + 10101111

31. 以补码形式执行下面每一个加法：

(a)10001100 + 00111001　　(b)11011001 + 11100111

32. 以补码形式执行下面每一个减法：

(a)00110011 - 00010000　　(b)01100101 - 11101000

33. 以补码形式用 11110001 乘以 01101010。

34. 以补码形式用 01000100 除以 00011001。

2.8 节　十六进制数

35. 把每一个十六进制数都转换为二进制数：

(a)38_{16}　(b)59_{16}　(c)$A14_{16}$　(d) $5C8_{16}$

(e)4100_{16}　(f)$FB17_{16}$　(g)$8A9D_{16}$

36. 把每一个二进制数都转换为十六进制数：

(a)1110　(b)10　(c)10111

(d)10100110　(e)1111110000　(f)100110000010

37. 把每一个十六进制数都转换为十进制数：

(a)23_{16}　(b)92_{16}　(c)$1A_{16}$　(d)$8D_{16}$

(e)$F3_{16}$　(f)EB_{16}　(g)$5C2_{16}$　(h)700_{16}

38. 把每一个十进制数都转换为十六进制数：

(a) 8　(b)14　(c)33　(d)52

(e)284　(f)2890　(g)4019　(h)6500

39. 执行下面的加法：

(a)$37_{16} + 29_{16}$　(b)$A0_{16} + 6B_{16}$　(c)$FF_{16} + BB_{16}$

40. 执行下面的减法：

(a)$51_{16} - 40_{16}$　(b)$C8_{16} - 3A_{16}$　(c)$FD_{16} - 88_{16}$

2.9 节　八进制数

41. 把下面每一个八进制数都转换为十进制数：

(a)12_8　(b)27_8　(c)56_8　(d)64_8　(e)103_8

(f)557_8　(g)163_8　(h)1024_8　(i)7765_8

42. 使用重复除以 8 的方法，把下面每一个十进制数都转换为八进制数：

(a)15　(b)27　(c)46　(d)70

(e)100　(f)142　(g)219　(h)435

43. 把下面每一个八进制数都转换为二进制数：

(a)13_8　(b)57_8　(c)101_8　(d)321_8　(e)540_8

(f)4653_8　(g)13271_8　(h)45600_8　(i)100213_8

44. 把下面每一个二进制数都转换为八进制数：

(a)111　(b)10　(c)110111

(d)101010　(e)1100　(f)1011110

(g)101100011001　(h)10110000011　(i)111111101111000

2.10 节　二－十进制码(BCD)

45. 把下面每一个十进制数都转换为 8421 BCD 码：

(a)10 (b)13 (c)18 (d)21 (e)25 (f)36
(g)44 (h)57 (i)69 (j)98 (k)125 (l)156

46. 把习题 45 中的每个十进制数都转换为直接的二进制数，比较两者所需要的位数。
47. 把下面的十进制数转变为 BCD：
(a)104 (b)128 (c)132 (d)150 (e)186
(f)210 (g)359 (h)547 (i)1051
48. 把下面每一个 BCD 数都转换为十进制数：
(a)0001 (b)0110 (c)1001
(d)00011000 (e)00011001 (f) 00110010
(g)01000101 (h)10011000 (i) 100001110000
49. 把每一个 BCD 数都转换为十进制数：
(a)10000000 (b)001000110111 (c)001101000110
(d)010000100001 (e)011101010100 (f)100000000000
(g)100101111000 (h)0001011010000011 (i)1001000000011000
(j)0110011001100111
50. 将下面的 BCD 数相加：
(a)0010 + 0001 (b)0101 + 0011 (c)0111 + 0010
(d)1000 + 0001 (e)00011000 + 00010001 (f)01100100 + 00110011
(g)01000000 + 01000111 (h)10000101 + 00010011
51. 将下面的 BCD 数相加：
(a)1000 + 0110 (b)0111 + 0101 (c)1001 + 1000 (d)1001 + 0111
(e)00100101 + 00100111 (f)01010001 + 01011000
(g)10011000 + 10010111 (h)010101100001 + 011100001000
52. 把下面每对十进制数都转换为 BCD，并且执行加法运算：
(a)4 + 3 (b)5 + 2 (c)6 + 4 (d)17 + 12
(e)28 + 23 (f)65 + 58 (g)113 + 101 (h)295 + 157

2.11 节 数字编码

53. 在某个应用中，一个 4 位二进制序列从 1111 到 0000 周期性地循环。有 4 个数位发生变化，由于电路延时，这些变化可能不在同一瞬间出现。例如，如果最低位(LSB)首先变化，该数将在从 1111 到 0000 转换期间，显示为 1110，并且可能被系统误认。阐述格雷码是怎样避免这个问题的。
54. 把每一个二进制数都转换为格雷码：
(a)11011 (b)1001010 (c)1111011101110
55. 把每一个格雷码都转换为二进制数：
(a)1010 (b)00010 (c)11000010001

2.12 节 数字检测和校验码

56. 确定下面哪一个偶校验码有错：
(a)100110010 (b)011101010 (c)10111111010001010
57. 确定下面哪一个奇校验码有错：
(a)11110110 (b)00110001 (c)01010101010101010
58. 给下面的每一个数据字节添加合适的偶校验位：
(a)10100100 (b)00001001 (c)11111110
59. 为数据位 1100 确定偶校验汉明码。

60. 为数据位 11001 确定奇校验汉明码。

61. 纠正下面的偶校验汉明码的错误:

(a)1110100 (b)1000111

62. 纠正下面的奇校验汉明码的错误:

(a)110100011 (b)100001101

自测题答案

1.(d) 2.(a) 3.(b) 4.(c) 5.(c) 6.(a) 7.(d) 8.(b) 9.(d) 10.(a) 11.(c) 12.(d) 13.(d) 14.(b) 15.(c) 16.(a) 17.(c) 18.(b)

第3章　逻　辑　门

章节提纲

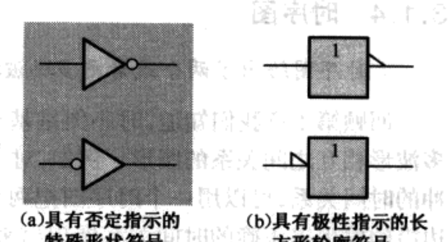
3.1　反相器

反相器(非门电路)进行称为反相或反码的运算。反相器把一种逻辑电平转变为相反的逻辑电平。就位而言,它把1变为0和把0变为1。

学完本节以后,应当能够

- 识别否定和极性指示
- 通过特殊形状符号或者矩形轮廓符号来识别反相器
- 为反相器写出真值表
- 描述反相器的逻辑运算

反相器的标准逻辑符号如图3.1所示。图3.1(a)给出了特殊形状符号,图3.1(b)给出了长方轮廓符号。在这本书中,一般使用特殊形状符号;但是,矩形轮廓符号常常出现在许多工业出版物中,所以也应该熟悉它们。(逻辑门符号依据ANSI/IEEE标准91-1984。)



(a)具有否定指示的特殊形状符号　(b)具有极性指示的长方形轮廓符号

图3.1　反相器的标准逻辑符号(ANSI/IEEE标准91-1984)

3.1.1　否定和极性指示

否定指示是一个小圆圈(○),当其出现在任何逻辑元件的输入或者输出位置时,都为反相或者为反码,如图3.1(a)的反相器所示。一般来说,输入位于逻辑符号的左边而输出位于右侧。当出现在输入位置时,小圆圈就表示0电平有效的或者是确定的输入状态,而这个输入称为低电平有效输入。当出现在输出位置时,该小圆圈就指明0电平有效的或者是确定的输出状态,而这个输出称为低电平有效输出。当输入或者输出没有小圆圈时,就表示1是有效的或者是确定的状态,而这个输入或输出称为高电平有效。

极性或者电平指示是一个“三角形”(◺),当其出现在任何逻辑元件的输入或者输出位置时,表示反相,如图3.1(b)所示。当其出现在输入位置时,就表示低电平是有效的或者是确定的输入状态。当其出现在输出位置时,就表示低电平是有效的或者是确定的输出状态。

两种指示(小圆圈或三角形)都可以用在特殊形状符号和矩形轮廓符号中。图 3.1(a)给出的是本书后面主要使用的反相器符号。注意反向或极性指示的不同放置并不意味着反相器运算方式的改变。

3.1.2 反相器真值表

当反相器输入高电平时,它的输出就是低电平。当反相器的输入是低电平时,它的输出就是高电平。这种运算总结于表 3.1 中,其中以电平和对应的位值给出了每个可能的输入和与之对应的输出。这样的表称之为真值表。

表 3.1 反相器真值表

输入	输出
LOW (0)	HIGH (1)
HIGH (1)	LOW (0)

3.1.3 反相器运算

图 3.2 给出了反相器脉冲输入和相应的输出,其中 t_1 和 t_2 指明了在输入和输出波形上相对应的点。

输入为低电平时,输出就是高电平;当输入是高电平时,输出就是低电平,因此产生反相的输出脉冲。

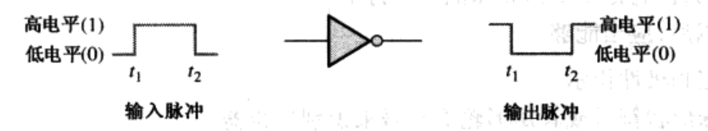


图 3.2 具有脉冲输入的反相运算

3.1.4 时序图

◇ 时序图给出了两个或者更多的波形在时间上的相互关系。

回顾第 1 章我们知道,时序图是基于时间的、准确显示两个或者更多波形相互之间关系的图形。例如,对于图 3.2 中的输出脉冲和输入脉冲的时间关系,可以用一个时序图把两个脉冲对准,从而使得这些脉冲边沿的发生以正确的时间关系展现出来。输入脉冲的上升沿和输出脉冲的下降沿在相同的时间出现(理想状态)。类似地,输入脉冲的下降沿和输出脉冲的上升沿也在相同的时间出现(理想状态)。这种时序关系如图 3.3 所示。时序图在说明具有多个脉冲的数字波形之间的时间关系时特别有用。

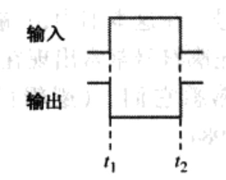


图 3.3 在图 3.2 条件下的时序图

例 3.1 图 3.4 中的反相器输入了一个波形。根据输入确定输出波形,并画出时序图。根据小圆圈的安放位置,有效输出状态是什么?

解: 输出波形精确地和输入波形相反(反相),如图 3.5 所示,该图是基础时序图。有效或者确定的输出状态是 **0**。

相关问题: 如果所给出的反相器否定指示(小圆圈)在输入上而不是在输出上,这会对时序图有什么影响?

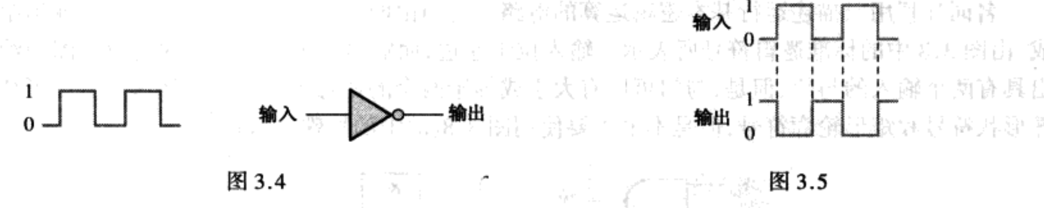


图 3.4 图 3.5

3.1.5 反相器的逻辑表达式

◇ 布尔代数使用变量和运算符来描述逻辑电路。

在布尔代数(它是逻辑电路的数学并且将在第4章中得到全面的介绍)中,变量由一个字母表示。变量的反码由字母上方的横杠来表示。变量可取的值是1或者0。如果一个变量是1,它的反码就是0,反之亦然。

反相器(非门电路)的运算可以用下面的方式表示:如果输入变量为 A,输出变量为 X,那么

$$X = \overline{A}$$

这个表达式说明输出是输入的反码,所以如果 $A = 0$,那么 $X = 1$;而如果 $A = 1$ 那么 $X = 0$。图3.6给出了这种情况。反码变量可以读做 A 横杠(A bar),或者 A 反(not A)。

3.1.6 应用举例

图3.7给出了一个产生8位二进制数反码的电路。二进制数的位被加到反相器的输入端,二进制数的反码出现在输出端。

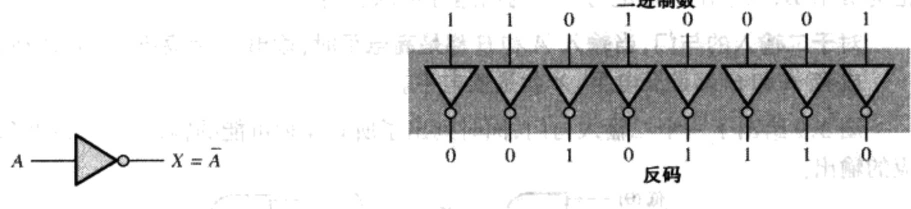


图 3.6 反相器对输入变量求反

图 3.7 使用反相器的反码电路举例

3.2 与门

与门(AND gate)是基本逻辑门之一,这些基本逻辑门可以组合在一起形成各种逻辑功能。与门可以有两个或者更多的输入,它的运算称之为逻辑乘。

学完本节以后,应当能够

- 通过特殊形状符号或者矩形轮廓符号来识别与门
- 描述与门的运算
- 为有任意输入数目的与门写出真值表
- 绘制具有任何指定波形的与门的时序图
- 为有任意输入的与门写出逻辑表达式
- 讨论与门应用的例子

名词“门”用以描述运行基本逻辑运算的电路。与门由两个或者更多的输入和一个输出组成,由图3.8中的标准逻辑符号所表示。输入位于左边,而输出位于每个符号的右边。图中给出具有两个输入的与门;但是,与门可以有大于或等于两个的任意个输入。虽然这里给出了特殊形状符号和矩形轮廓符号,但是本书主要使用图3.8(a)中的特殊形状符号。

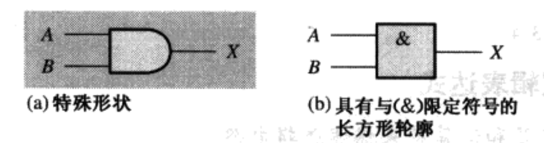


图3.8 与门的标准逻辑符号,给出了两个输入(ANSI/IEEE标准91-1984)



计算机小知识

逻辑门是计算机的构件。除了某种特定类型的存储器,计算机中的大多数功能都由使用广泛的逻辑门来实现。例如,计算机的主要部件微处理器,就是由几十万个甚至上百万个逻辑门组成的。

3.2.1 与门运算

◇ 与门可以具有多于两个的输入。

当且仅当与门所有的输入是高电平时,才会输出高电平。当任何一个输入为低电平时,输出就是低电平。所以,与门的基本用途是判断若干条件是否同时为真,为真时所有的输入是高电平,并且在输出产生高电平以表示所有的条件都为真。图3.8中的两输入与门的输入被标记为 A 和 B,而输出被标记为 X。与门运算可以表述为

对于二输入的与门,当输入 A 和 B 都是高电平时,输出 X 为高电平;当 A 或 B 是低电平,或者 A 和 B 都是低电平,X 就是低电平。

图3.9给出了一个二输入与门,同时列出了所有4种可能的输入组合,以及每个与门相对应的输出。

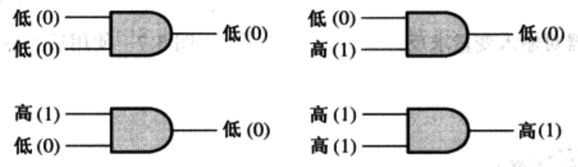


图3.9 二输入与门的所有可能逻辑电平

3.2.2 与门真值表

◇ 对于一个与门来说,输入都是高电平时才会产生高电平输出。

与门的逻辑运算可以用真值表来表示,真值表列出了所有的输入组合及相应的输出。如表3.2的二输入与门所示。该真值表可以扩展到任意个数的输入。虽然名词高电平和低电平往往指“物理”意义的输入和输出状态,不过真值表给出的却

表3.2 二输入与门的真值表

输入		输出
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

1 = 高电平, 0 = 低电平

是1和0;在正逻辑中,高电平就相当于1,而低电平就相当于0。对于任意的与门,不管有多少个输入,当且仅当所有的输入为高电平时,输出才是高电平。

逻辑门输入的所有二进制组合的总数,由下面的公式确定:

$$N = 2^n \tag{3.1}$$

这里,N是可能输入组合的个数,n是输入变量的个数。说明如下:

- 二输入变量:$N = 2^2 = 4$ 种组合
- 三输入变量:$N = 2^3 = 8$ 种组合
- 四输入变量:$N = 2^4 = 16$ 种组合

使用式(3.1),可以为具有任意多个输入的门确定输入位组合的个数。

例 3.2

(a)为三输入与门建立真值表;

(b)为四输入与门确定可能输入组合的总数。

解:

(a) 三输入与门,有8个可能的输入组合($2^3 = 8$)。真值表(见表3.3)的输入一边给出了三位二进制数的所有8种组合。除了3个输入数位都是1的情况,输出一边全是0。

(b) $N = 2^4 = \mathbf{16}$。对于四输入与门,有16种可能的二进制输入数位组合。

相关问题:为四输入与门开发真值表。

3.2.3 波形输入运算

在多数应用实例中,门的输入电平不是固定的,而是在逻辑高电平和逻辑低电平之间频繁变化的电压波形。现在看看具有脉冲波形输入的与门运算,要记住与门遵循真值表的运算,而不用考虑输入电平是不变的还是高低来回变化的电平。

表 3.3

输入			输出
A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

检查一下与门的波形运算,通过观察相互关联的输入以确定任意给定时刻下的输出电平。在图3.10中,在时间间隔t_1期间,输入A和B都是高电平(1),因此在该期间的输出X为高电平(1)。在时间间隔t_2期间,输入A为低电平(0)而输入B为高电平(1),所以输出是低电平。在时间间隔t_3期间,输入又都是高电平(1),所以输出是高电平(1)。在时间间隔t_4期间,输入A是高电平(1)而输入B是低电平(0),因此产生一个低电平(0)。最后,在时间间隔t_5期间,输入A为低电平(0),输入B是低电平(0),所以输出就是低电平(0)。正如所知,输入和输出的波形图给出了一种时间关系,称为时序图。

例 3.3 如图3.11所示,如果两个输入波形A和B加到与门的输入,那么输出波形的结果是什么?

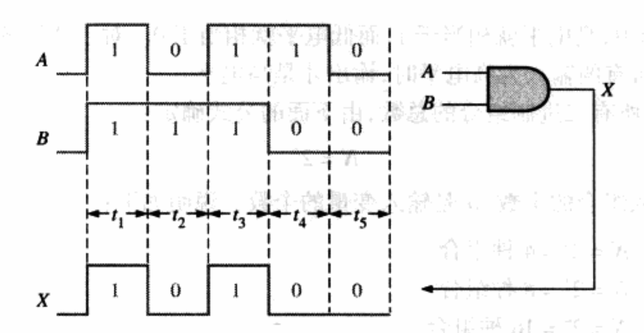


图 3.10 具有时序图的与门举例,图中给出了输入输出的关系

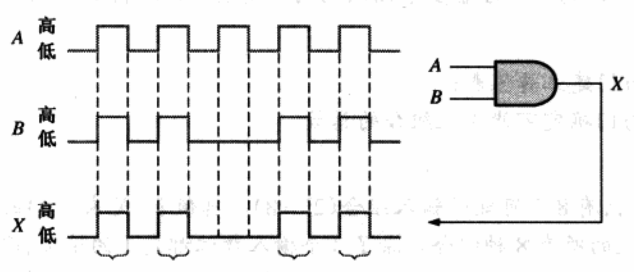


图 3.11

解: 如图 3.11 中的时序图所给出的那样,当且仅当 *A* 和 *B* 波形都是高电平时,输出波形 *X* 才是高电平。

相关问题: 如果图 3.11 中波形 *A* 的第 2 个和第 4 个脉冲改为低电平,确定输出波形并给出时序图。

记住,在分析逻辑门的波形运算时,注意输入之间的时间关系及输入和输出之间的时间关系是非常重要的。

例 3.4 在图 3.12 中,根据两个输入波形 *A* 和 *B*,利用输出和输入之间的相应关系,给出输出波形。

解: 只有时序图中两个输入波表都是高电平,输出波形才是高电平。

相关问题: 如果图 3.12 中与门的 *B* 输入总是高电平,确定输出波形。

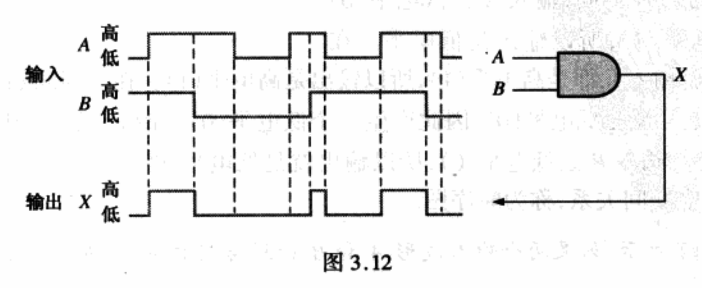


图 3.12

例 3.5 对于图 3.13 中的三输入与门,确定输出波形和输入波形之间的关系。

解：当所有的3个输入波形 A、B 和 C 都是高电平时，三输入与门的输出波形 X 才是高电平。

相关问题：如果 C 输入总是高电平，那么图3.13中与门的输出波形是什么？

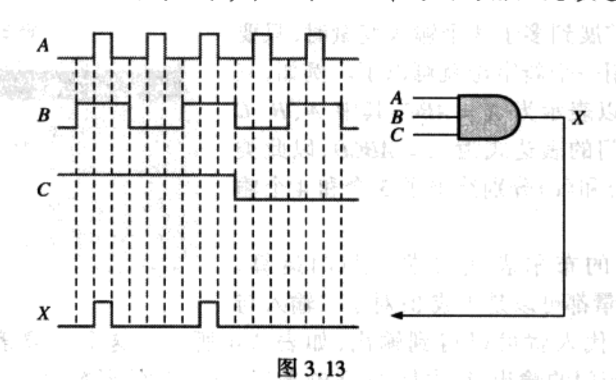


图3.13

3.2.4 与门的逻辑表达式

两个变量的逻辑与函数在数学上可以表示为：在两个变量之间放一个点，如 $A \cdot B$；或不使用点，仅仅写出两个相邻的字母，如 AB。因为方便，通常使用后面的写法。

布尔乘法 遵循二进制乘法相同的基本法则，在第2章已经讨论过了，如下所示：

$0 \cdot 0 = 0$
$0 \cdot 1 = 0$
$1 \cdot 0 = 0$
$1 \cdot 1 = 1$

布尔乘法和与函数相同。

二输入与门的运算可以表示为如下的等式形式；如果一个输入变量是 A，另一个是 B，输出变量是 X，那么布尔表达式就是

$$X = AB$$

图3.14(a)给出了二输入变量，并标出输出变量的与门逻辑符号。

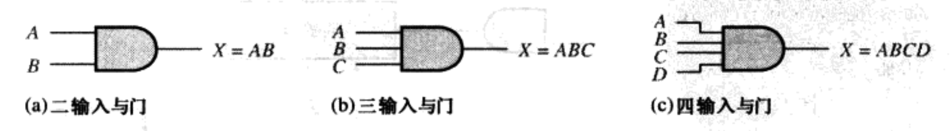


图3.14 具有2个、3个和4个输入的与门布尔表达式



计算机小知识

当需要选择操作一个或者多个数据字节的某些位时，计算机可利用所有的基本逻辑运算。利用掩码可以执行选择性位操作。例如，为了清除(使之变为0)某个数据字节的右4位，而保持左4位不变，可以把该数据

字节和11110000进行与运算来完成这个任务。注意每一个被0与的位都会变成0,而被1与的位保持不变。如果10101010和11110000进行与运算,结果就是10100000。

◇ 当变量 *ABC* 写在一起时,就是表示与运算。

要把与表达式扩展到多于两个输入变量时,只要为每个输入变量使用一个新字母就可以了。例如,三输入与门的函数可以表示为 $X = ABC$,其中 A、B、C 是输入变量。四与门的表达式为 $X = ABCD$,以此类推。图3.14中的(a)和(c)分别给出了3个和4个输入变量的与门。

表 3.4

A	B	$AB = X$
0	0	$0 \cdot 0 = 0$
0	1	$0 \cdot 1 = 0$
1	0	$1 \cdot 0 = 0$
1	1	$1 \cdot 1 = 1$

可以使用输出的布尔表达式求得与门运算。例如,输入的每个变量都可以是1或0;对于二输入与门,把公式 $X = AB$ 代入就可以得到输出,如表3.4所示。这个计算表明仅当两个输入都是1(高电平)时,与门的输出 X 才是1(高电平)。可以对任意数目的输入变量做相似的分析。

3.2.5 应用举例

与门作为使能/禁止设备 与门的一种常见应用是使能(也就是允许)信号(脉冲波形)在某个时间从一点传到另一点,并禁止(阻止)在其他时间传送。

图3.15给出了与门这种特殊用途的一个简单例子,其中与门用来控制信号(波形 A)向数字计数器的传送。这个电路的目的是测量波形 A 的频率。使能脉冲有1秒的精确宽度:当使能脉冲是高电平时,波形 A 就会通过与门到达计数器;而当使能脉冲是低电平时,不允许(禁止)信号通过。

在使能脉冲的1秒时间间隔内,波形 A 中的脉冲通过与门到达计数器,脉冲在1秒的时间间隔内通过的次数等于波形 A 的频率。例如,图3.15表示了一秒内有6个脉冲,也就是频率为6 Hz。如果在使能脉冲的1秒时间间隔期内,有1000个脉冲通过与门,这样就是1000脉冲/秒,或者是1 kHz的频率。

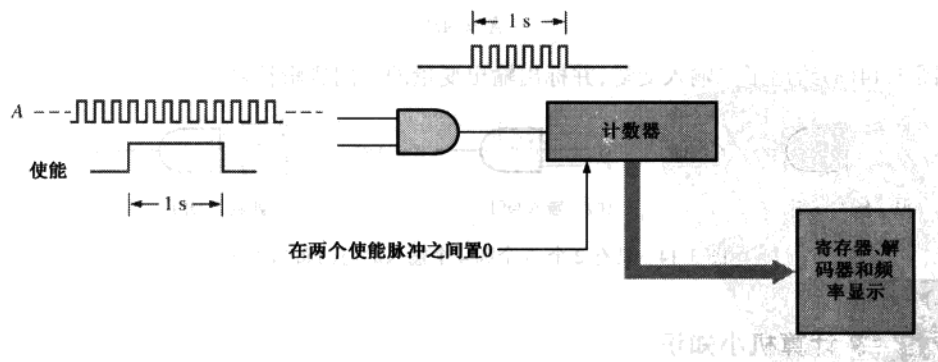


图3.15 频率计数器运行使能/禁止功能的一个与门

计数器记录每秒钟脉冲的个数,并产生一个二进制输出,进入解码和显示电路,产生可以

读取的频率。使能脉冲在一定的间隔期间内重复,一个新的计数值就会产生,所以如果频率变化,就会显示出新的数值。在两个使能脉冲之间,计数器被置为0,因而在每个使能脉冲到来时,计数器从0开始计数。当前的频率计数值保存在一个寄存器中,这样显示值不会受计数器的影响。

安全带警报系统 在图3.16中,与门用在了一个简单的汽车安全带警报系统中,用来检测是否点火开关已开,以及安全带是否解开。如果点火开关处于开的状态,就会产生一个高电平。如果安全带没有系好,与门的输入 B 上就产生一个高电平。同样,当点火开关打开时,计时器就会启动并且在输入 C 上产生一个30秒的高电平。所有的这三个条件都存在,也就是如果点火开关处于开的状态、安全带没有系好和计时器正在计时,与门的输出就是高电平,这时音响警报就会被激活以提醒司机。

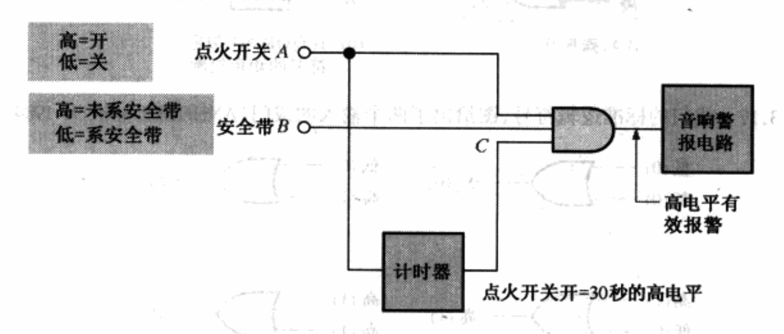


图3.16 使用与门的简单安全带警报电路

3.3 或门

或门(OR gate)是另一个基本门,这些基本门可以构建所有的逻辑功能。或门可以拥有两个或多个输入,它的运算称为逻辑加法。

学完本节之后,应当能够

- 通过特殊形状符号或者矩形轮廓符号识别或门
- 描述或门的运算
- 为具有任意多个输入的或门写出真值表
- 为具有特定输入波形的或门画出时序图
- 为具有任意多个输入的或门写出逻辑表达式
- 讨论或门应用的例子

◇ *或门可以具有多于两个的输入。*

或门具有两个或者更多的输入及一个输出,如图3.17中的标准逻辑符号给出了具有两个输入的或门。或门可以具有多于一个的任意输入。虽然同时给出了特殊形状和矩形轮廓符号,但是本书使用的是特殊形状符号。

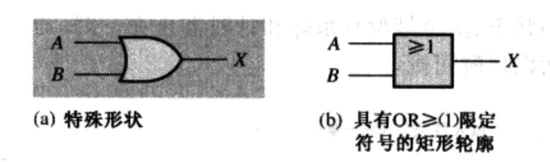
3.3.1 或门的运算

当任意一个输入为高电平时,或门的输出为高电平。当且仅当所有的输入是低电平时,输

出才会是低电平。所以,或门用来判断它的输入是否有一个或者多个高电平,有高电平输入时,输出一个高电平以表明条件满足。图 3.17 中二输入或门的输入被标以 A 和 B,输出则被标以 X,或门的运算可以做如下表述:

对于一个二输入或门来说,如果输入 A 和输入 B 中有一个是高电平,或者两者都为高电平,输出 X 就为高电平;仅当 A 和 B 都是低电平,X 为低电平。

高电平是或门的有效或者肯定输出电位。图 3.18 给出了二输入或门的所有 4 种可能输入组合的输入/输出结果。



(a) 特殊形状　　(b) 具有OR≥(1)限定符号的矩形轮廓

图 3.17　或门的标准逻辑符号,图给出了两个输入的或门(ANSI/IEEE 标准 91-1984)

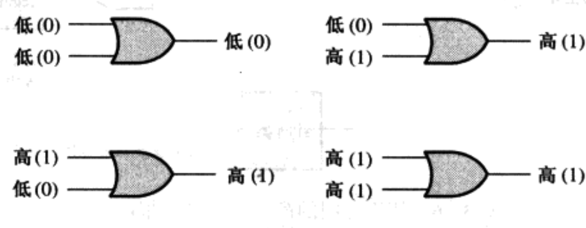


图 3.18　二输入或门的所有可能逻辑电位

3.3.2　或门真值表

◇ 对于或门来说,至少需要一个高电平输入才能产生高电平输出。

表 3.5 描述了二输入或门的运算。这个真值表可以扩展到任意个数的输入;但是不管有多少输入,当有一个或者多个输入是高电平时,输出就是高电平。

表 3.5　二输入或门的真值表

输入		输出
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

1 = 高电平, 0 = 低电平

3.3.3　波形输入的运算

现在观察具有波形输入的或门运算,记住它的逻辑运算。再有,分析具有脉冲波形的或门运算的关键是涉及的所有波形的时间关系。例如,在图 3.19 中,在时间间隔 t_1 期间,A 和 B 都是高电平 1,所以输出 X 为高电平 1。在时间间隔 t_2 期间,输入 A 是低电平 0,但是输入 B 是高电平 1,所以输出 X 为高电平 1。在时间间隔 t_3 期间,两个输入都是低电平 0,所以输出 X 为低电平 0。在时间间隔 t_4 期间,两个输入都是高电平 1,所以输出 X 为高电平 1。

在这个说明中,或门的真值表运算应用到了电平没有改变的时间间隔上。例 3.6 到例 3.8 将进一步说明波形输入的或门运算。

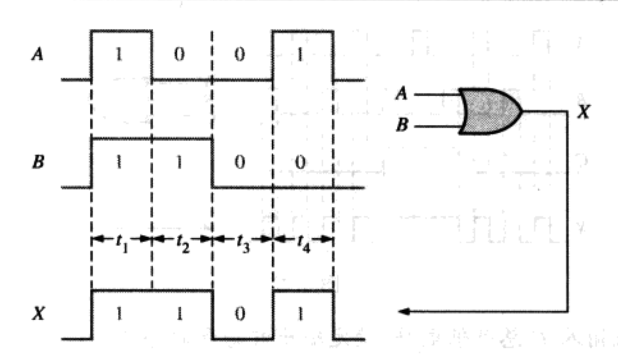


图 3.19 或门运算的时序图例子,图给出了输入和输出之间的时间关系

例 3.6 在图 3.20 中,如果两个输入波形 A 和 B 加到或门的输入,那么输出波形是怎样的呢?

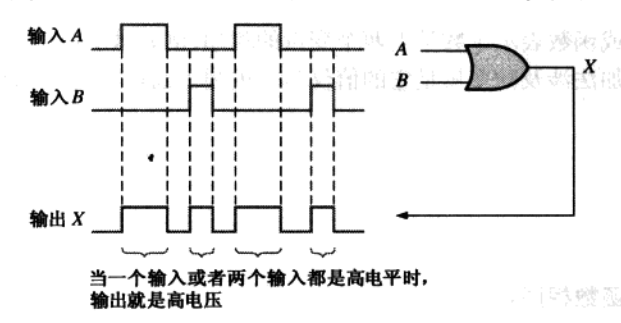


图 3.20

解: 如时序图所示,当一个输入或者两个输入都是高电平时,二输入或门的输出波形就是高电平。以上情况是,两个输入波形在相同的时刻不同时为高电平。

相关问题: 如果输入 A 做如下变化:从已有的第一个脉冲的开始到已有的第二个脉冲的结束处都为高电平,确定输出波形和给出时序图。

例 3.7 在图 3.21 中,对于两个输入波形 A 和 B,给出输出波形和输出与输入的对应关系。

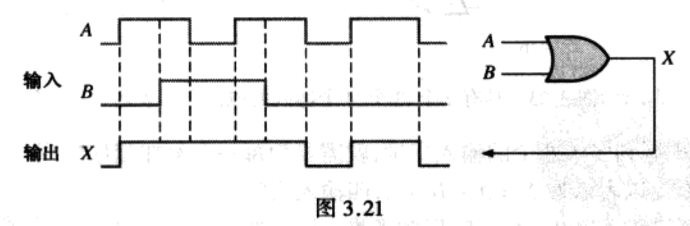


图 3.21

解: 如给出的时序图中的输出波形所示,当一个或两个输入为高电平时,输出为高电平。

相关问题: 如果以上输入 A 的中间脉冲由低电平取代,确定波形和画出时序图。

例 3.8 对于图 3.22 中的三输入或门,按照输入波形的时序关系画出其输出波形。

解: 当一个输入或者两个输入都是高电平时,或门的输出就是高电平。如时序图中的输出波形 X 所示。

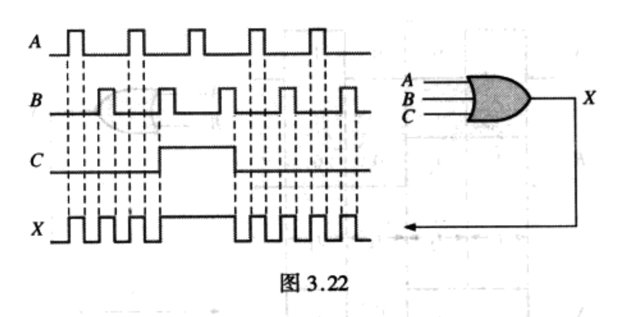


图 3.22

相关问题:如果输入 C 总是低电平,确定输出波形和时序图。

3.3.4 或门的逻辑表达式

◇ 当两个变量之间由 + 分开时,那么它们是或的关系。

二变量的逻辑或函数表示了数学上两个变量的相加,例如 $A+B$。

布尔代数中的加法涉及的变量是它的值仅为二进制 1 或 0。布尔代数的基本法则如下:

$0+0=0$

$0+1=1$

$1+0=1$

$1+1=1$

布尔加法和或函数相同。

注意,布尔加法和二进制加法的不同点在于两个 1 相加的情况。在布尔代数中没有进位。

二输入或门的运算可以做如下表述:如果一个输入变量是 A,另一个输入变量是 B,输出变量是 X,那么布尔代数的表达式就是

$$X=A+B$$

图 3.23(a)给出了二变量的或门逻辑符号,同时标出了输入和输出变量。

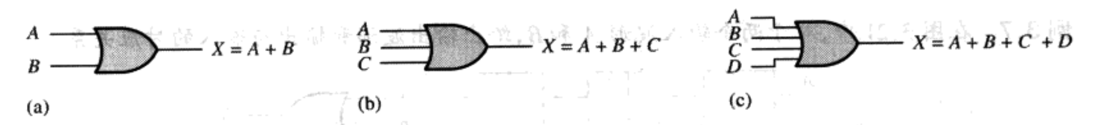


图 3.23 具有 2 个、3 个、4 个输入的或门布尔表达式

把或表达式扩展到多于两个的输入变量,就需要为每一个附加变量使用一个新字母。例如,三输入或门的函数可以表示为 $X=A+B+C$,四输入或门的表达式可以写成 $X=A+B+C+D$,以此类推。图 3.23 中的(b)和(c)分别给出了具有 3 个和 4 个变量的或门。

或门的运算可以由布尔表达式确定,把输入变量的所有 1 和 0 的可能组合代入输入变量,就可以得到输出 X,如表 3.6 的二输入或门所示。这个运算表明,当一个或两个输入是高电平时,输出 X 为高电平。相同的分析可以扩展到任意多个输入变量的或门。

表 3.6 二输入或门

A	B	$A+B=X$
0	0	$0+0=0$
0	1	$0+1=1$
1	0	$1+0=1$
1	1	$1+1=1$



计算机小知识

另一种应用于计算机编程中的掩码运算是:有选择性地使得一个数据字节中的某些位为1(称为置位),而不影响任何其他的位,这是由或运算来完成的。使用掩码使得被置位的数据位的任何位置上都是1。例如,如果想迫使某个字节的最高有效位等于1而其他位保持不变,就可以对数据字节和掩码10000000进行或运算。

3.3.5 应用举例

一个入室盗窃检测和警报系统的部分简化图如图3.24所示。这个系统可以用在的一个房屋中,即具有两扇窗户和一扇门的房间。传感器是磁性开关,它被打开时产生一个高电平输出,关闭时产生一个低电平输出。只要窗户和门是安全的,开关就是关着的并且或门的三个输入都是低电平。当一扇窗户或者门被打开时,在或门的输入端就会产生一个高电平,输出端就是高电平。然后激活和锁存警报电路,以发出入侵警报。

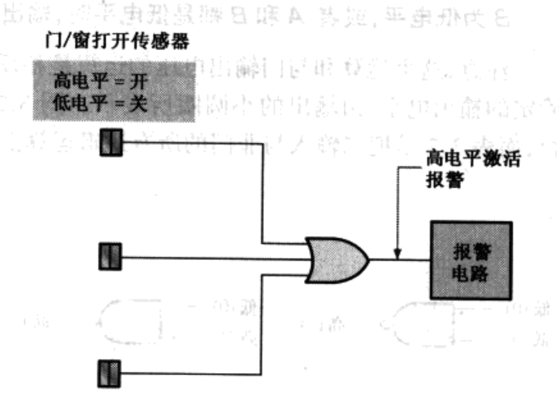


图3.24 使用或门的一个简化入室盗窃检测和警报系统

3.4 与非门

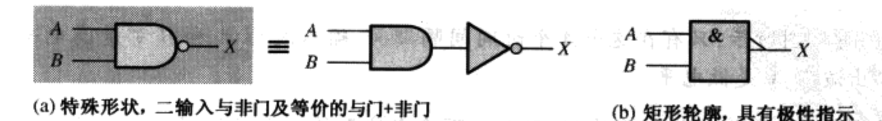
与非门(NAND gate)是一种很常用的逻辑元件,因为它可以用做通用门;也就是说,可以用它的组合来完成与、或和反相运算。与非门的通用特性将会在第5章得到完整的验证。

学完本节之后,应当能够

- 通过特殊形状符号或者矩形轮廓符号来识别与非门
- 描述与非门的运算
- 为具有任意多个输入的与非门写出真值表
- 为具有任意特定输入波形的与非门画出时序图
- 为具有任意多个输入的与非门写出逻辑表达式
- 描述非-或等价的与非门运算
- 讨论与非门应用的例子

◇ 除了输出被反相以外,与非门和与门是一样的。

单词NAND是NOT-AND的缩写,意思是具有反码(反相)输出的与函数。二输入的与非门的标准逻辑符号和一个与门其后再加一个反相器的电路图等价,如图3.25(a)所示,其中符号≡是指等价于。图3.25(b)给出了矩形轮廓符号。



(a) 特殊形状,二输入与非门及等价的与门+非门

(b) 矩形轮廓,具有极性指示

图3.25 标准与非门逻辑符号(ANSI/IEEE 标准 91-1984)

3.4.1 与非门的运算

只有所有的输入是高电平时,与非门才会输出其低电平。当任何一个输入为低电平时,输出就是高电平。二输入与非门的一个具体例子如图 3.25 所示,其中输入被标为 A 和 B,输出被标为 X,该运算可以做如下表述:

对于一个二输入与非门,当输入 A 和 B 都是高电平时,输出 X 就是低电平;当输入 A 或 B 为低电平,或者 A 和 B 都是低电平时,输出 X 就是高电平。

注意,这个运算和与门输出电压的运算是相反的。在一个与非门中,低电平(0)是有效或确定的输出电平,由输出的小圆圈所表示。图 3.26 说明了二输入与非门的所有 4 种可能组合,而表 3.7 是把二输入与非门的所有逻辑运算汇总后的真值表。

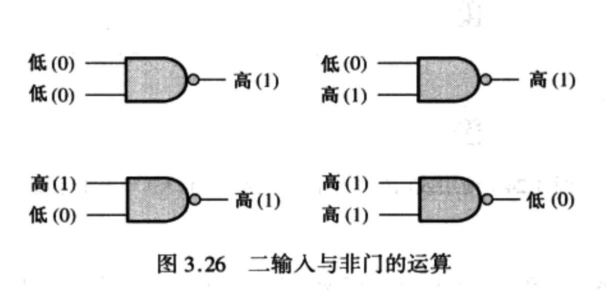
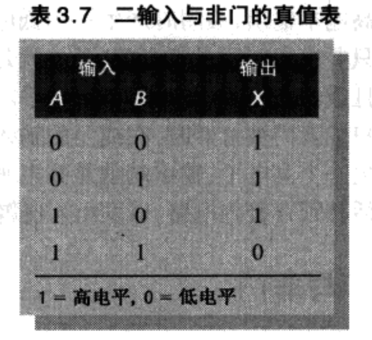


图 3.26 二输入与非门的运算

表 3.7 二输入与非门的真值表



输入		输出
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

1 = 高电平, 0 = 低电平

3.4.2 波形输出运算

现在来看与非门的波形运算。从真值表可以知道,仅当所有的输入为高电平时,输出才是低电平。

例 3.9 图 3.27 中的两个输入波形 A 和 B 加到与非门的输入,请确定输出波形。

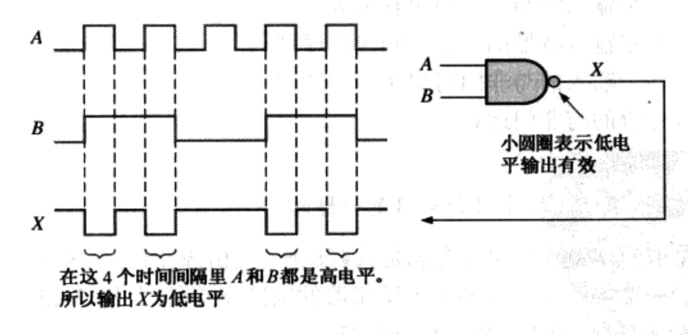


图 3.27

解: 如时序图所示,只有在这个 4 个时间间隔期间,输入波形 A 和 B 都是高电平,与非门的输出波形 X 是低电平。

相关问题: 如果输入波形 B 反相,确定一下输出波形并画出时序图。

例 3.10 在图 3.28 中,按照输入波形的时序关系画出三输入与非门的输出波形。

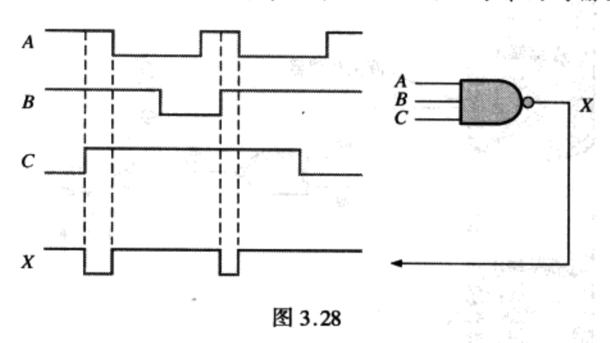


图 3.28

解:只有当所有的输入波形都是高电平时,与非门的输出波形 X 才是低电平,如时序图所示。

相关问题:如果输入波形 A 反相,确定画出输出波形和时序图。

与非门的非 - 或等价运算 与非门运算的内在特性是这样的:一个或者多个低电平输入产生一个高电平输出。表 3.7 给出了当任何一个输入 A 和 B 是低电平 0 时,输出就是高电平 1。从这个观点来说,与非门可以用于需要一个或者多个低电平输入并且产生高电平输出的或运算。与非门运算的这个功能称为非 - 或运算。词语"非"在本文中的意思是当输入为低电平时,输入就被定义为处于有效或者确定状态。

对于二输入与非门执行非 - 或运算,如果输入 A 或者 B 是低电平,或者 A 和 B 都是低电平时,输出 X 就是高电平。

当与非门用以检测一个或者多个低电平输入而不全是高电平输入时,它就是在执行非或运算,并且由图 3.29 中的标准逻辑符号表示出来。虽然图 3.29 中的两个符号表示相同的物理门,但这是在特定的应用中用来定义的逻辑门或者运算模式,如例 3.11 到例 3.13 所示。

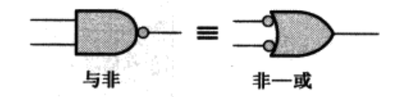


图 3.29 表示与非门的两个等效运算的标准符号

例 3.11 一个制造工厂使用两个存储罐保存在制造加工过程中需要的某种液体化学物质。每一个存储罐都有一个传感器用来检测什么时候化学物质液位会降到满罐的 25%。当储罐储量大于 1/4 满罐时,传感器就产生一个 5 V 电压。当罐中的化学物质体积降至 1/4 满罐时,传感器就会输出 0 V 电压。

指示器面板上需要一个绿色发光二极管(LED),用来显示什么时候两个存储罐的储量都大于 1/4 的满罐。给出如何用与非门实现这个功能的方法。

解:图 3.30 给出了具有两个输入的与非门,此与非门的输入和存储罐液位传感器连接,输出连接到指示器面板上。所完成的功能可以表述为:如果存储罐 A 和 B 的储量都达 1/4 的满罐以上,发光二极管就会点亮。

只要两个传感器输出都是高电平(5 V),也就是表示两个储罐的储量都大于 1/4 的满罐,那么与非门的输出就是低电平(0 V)。设计绿色发光二极管电路使得低电平点亮。

相关问题:若要监测 3 个储罐而不是 2 个储罐,应当对图 3.30 中的电路做怎样的修改?

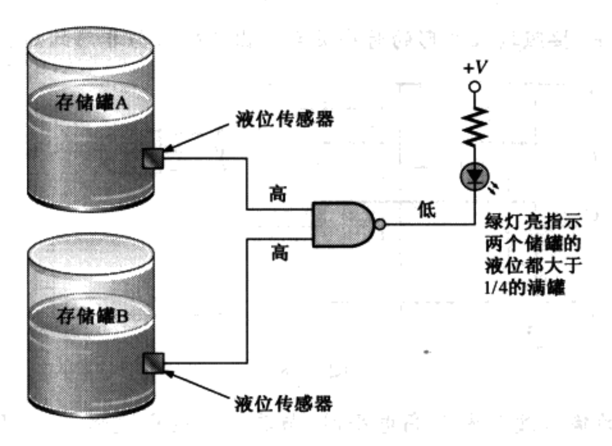


图 3.30

例 3.12 例 3.11 中所描述的生产过程的监督员,决定使用一个红色发光二极管显示下面的情况:只要有一个存储罐的液为降至 1/4 的满罐时就点亮,替代绿色发光二极管点亮指示两个存储罐液面都高于 1/4 的满罐的情况。给出如何实现这个功能的方法。

解: 图 3.31 给出了一个非 - 或门符号的与非门,用以检测至少有一个低电平输入的情况。如果存储罐内体积达到 1/4 的满罐或更低时,传感器就输出一个低电平。发生这种情况时,非 - 或门就会输出高电平,这个高电平使得面板上的电路把红色发光二极管点亮。所完成的功能可以表述为:如果存储罐 A 或者 B 或者 A 和 B 同时低于 1/4 的满罐,发光二极管点亮。

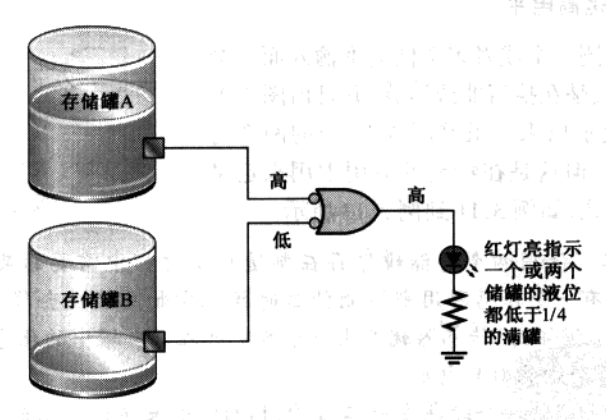


图 3.31

注意:这个例子和例 3.11 中一样,使用了相同的二输入与非门,但是在图表中使用了不同的符号,说明了与非门和等效的非 - 或门运算在不同方式下的使用。

相关问题: 怎样修改图 3.31 中的电路才能监测 4 个存储罐而不是 2 个存储罐?

例 3.13 对于图 3.32 中的四输入与非门,这里看做非 - 或门的运算,根据输入确定输出。

解: 只要有一个输入是低电平,输出波形 X 就是高电平,如时序图所示。

相关问题: 如果输入波形 A 在应用于与非门之前被反相,请确定输出波形。

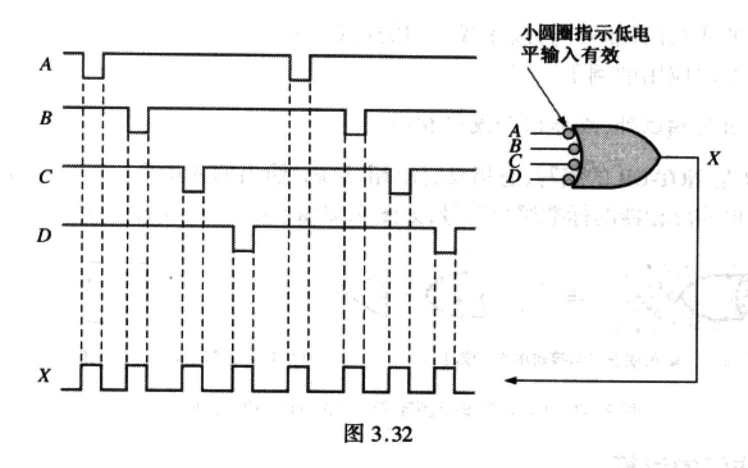


图 3.32

3.4.3 与非门的逻辑表达式

二输入与非门输出的布尔表达式为

$$X = \overline{AB}$$

变量上方的横杠就表示反相。这个表达式指出，两个变量 A 和 B 首先进行与运算然后再取反，这由与运算表达式上的横杠来表示。这就是二输入与非门运算的方程表示形式。两个输入变量所有可能的值通过此方程可以得到输出，结果如表 3.8 所给出。

表 3.8

A	B	$\overline{AB} = X$
0	0	$\overline{0 \cdot 0} = \overline{0} = 1$
0	1	$\overline{0 \cdot 1} = \overline{0} = 1$
1	0	$\overline{1 \cdot 0} = \overline{0} = 1$
1	1	$\overline{1 \cdot 1} = \overline{1} = 0$

一旦给定的逻辑函数的表达式确定下来，对应于变量所有可能的值，通过函数都可以运算出结果。对于每一种输入组合，运算结果会准确地给出逻辑电路的输出。因此，就对这个电路的逻辑运算做出了完整的描述。与非门表达式可以扩展到多于两个的输入变量，扩展的变量可以用其他的字母来表示。

3.5 或非门

或非门(NOR gate)和与非门相似，也是很有用的逻辑元件，因为它同样可以用做通用门：也就是说，或非门可以通过组合来完成与、或和反相的运算。或非门的通用特性将在第 5 章得到完整的验证。

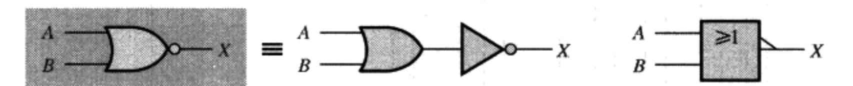
学完本节以后，应当能够

- 通过特殊形状符号或者矩形轮廓符号识别或非门
- 描述或非门的运算
- 为有任意数目输入的或非门写出真值表
- 为有任意指定波形的或非门画出时序图
- 为有任意数目输入的或非门写出逻辑表达式

- 用等价的非与门(非 - 与)表述或非门的运算
- 讨论或非门应用的例子

◇ 除了输出反相以外,或非门和或门相同。

词汇 NOR 是 NOT-OR 的缩写,意指具有反相(反码)输出的或运算。二输入或非门和与之等价的或门后面再加反相器的标准逻辑符号,如图 3.33(a)所示。图 3.33(b)给出了矩形轮廓符号。



(a)特殊形状,二输入或非门和等价的非-或门 (b)长方形轮廓,具有极性指示的二输入或非门

图 3.33 标准或非门逻辑符号(ANSI/IEEE 标准 91-1984)

3.5.1 或非门的运算

当任何一个输入为高电平时,或非门就输出一个低电平。只有当所有的输入是低电平时,输出才是高电平。对于二输入或非门的情况,如图 3.33 所示,输入被标为 A 和 B,输出被标记为 X,该运算可以做如下表述:

对于二输入或非门,如果输入 *A* 或者输入 *B* 为高电平,或者 *A* 和 *B* 都是高电平;输出 *X* 就是低电平;如果 *A* 和 *B* 都是低电平,输出 *X* 就是高电平。

这个运算产生和或门相反的输出电平。在或非门中,低电平输出是有效的或者是确定的输出电平,由输出的小圆圈表示。图 3.34 说明了二输入或非门的运算,包括所有可能的 4 种输入组合,而表 3.9 是二输入或非门的真值表。

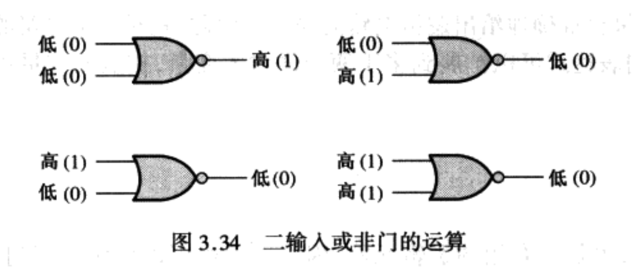
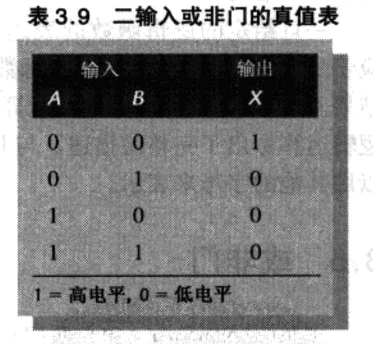


图 3.34 二输入或非门的运算

表 3.9 二输入或非门的真值表



输入		输出
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

1 = 高电平, 0 = 低电平

3.5.2 波形输入运算

下面的两个例子给出了具有脉冲波形输入的或非门运算。和其他类型的门一样,只要遵循真值表就可以确定输出波形和输入的正确时间关系。

例 3.14 如果图 3.35 中的两个波加到或非门的输入,那么输出波形是什么?

解: 无论什么时候或非门的输入为高电平时,输出就是低电平,如时序图中的输出波形 X 所示。

相关问题: 把输入 B 反相,确定输出波形和输入波形之间的关系。

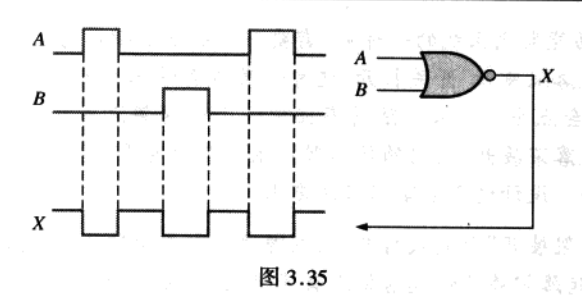


图 3.35

例 3.15 给出图 3.36 中三输入或非门的输出波形和输入波形之间的正确时间关系。

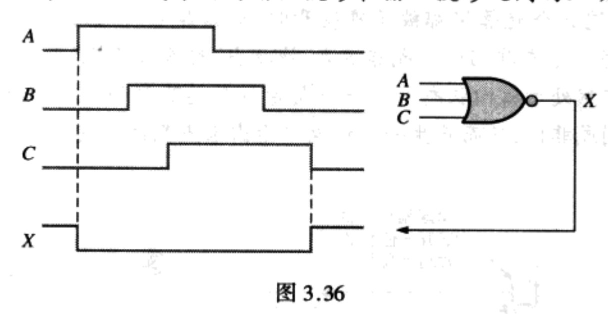


图 3.36

解:当任何一个输入为高电平时,输出 X 就是低电平,如时序图中的输出波形 X 所示。

相关问题:在输入 B 和 C 反相的情况下,确定输出并画出时序图。

或非门的非-与等价运算 或非门和与非门相似,并具有其运算的另一个特点,也就是固有的逻辑函数功能。表 3.9 给出了只有当所有的输入都是低电平时,才会产生高电平输出。从这点来说,或非门可以用做需要所有的低电平输入来产生高电平输出的与运算。或非门的这个功能称为非-与。这里词语"非"意指低电平输入有效或低电平为确定状态。

对非-与运算的二输入或非门来说,如果输入 A 和 B 都是低电平时,输出 X 就是高电平。

当一个或非门用来检测所有的低电平输入而不是一个或者多个高电平时,它就是在执行非-与运算,这由图 3.37 中的标准符号表示。图 3.37 中的两个符号表示相同的物理门并且仅用于区分两种运算模式,记住这一点很重要。下面的三个例子给出了说明。

例 3.16 当两个低电平出现在输入端并且输出高电平结果时,就需要一个设备进行指示。给出此设备。

解:当两个输入都是低电平并且输出高电平时,就需要使用二输入或非门,并把它看做非-与门,如图 3.38 所示。

相关问题:当有一个或者两个高电平输入并且产生低电平输出结果时,就需要一个设备进行指示,给出这个设备。

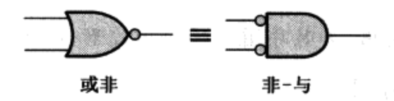


图 3.37 表示或非门两种等价运算的标准符号

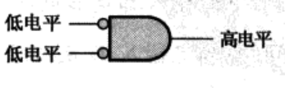


图 3.38

例 3.17 作为飞机功能监测系统的一部分，需要一个电路来指示着陆之前起落架的状态。当准备着陆时，“起落架展开”开关打开，这时如果所有的三个起落架都正确展开，绿色发光二极管(LED)就会点亮。如果着陆前有任何一个起落架没有正确展开，那么红色发光二极管被点亮。起落架展开时，它的传感器产生一个低电平。当起落架收回时，传感器就会产生一个高电平。设计这个电路以满足需求。

解：只有在“起落架展开”开关被打开时，电源才会加在电路上。如图 3.39 所示，对应于以上的两种情况(起落架展开和起落架没有正确展开)分别使用一个或非门以满足要求。其中一个或非门用做非－与门，用以检测三个起落架传感器的低电平。当所有三个门都是低电平，也就是这三个起落架都被正确展开时，那么非－与门的输出高电平就会点亮绿色发光二极管。另一个或非门运行或非运算，检测当“起落架展开”开关打开时，是否有一个或者多个起落架仍然处于收回状态。当一个或者多个起落架仍然处于收回状态时，传感器的高电平就会传到或非门，从而产生一个低电平输出来点亮红色发光二极管。

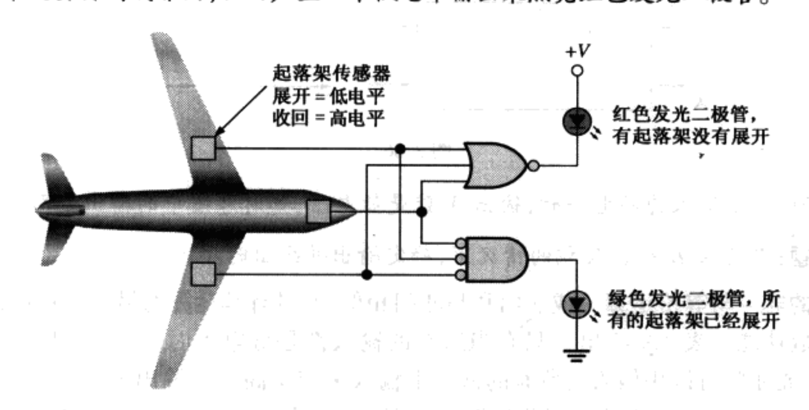


图 3.39

相关问题：在飞机起飞之后，应当使用什么类型的门来检测是否所有的三个起落架都被收回，假设需要一个低电平输出去点亮发光二极管。

相关提示

当使用一个逻辑门去驱动如发光二极管这样的负载时，查询一下生产厂商的数据资料，以得到逻辑门的最大驱动能力(输出电流)。通常一个集成逻辑门可能不能输出足够的电流，驱动如发光二极管这样的负载。许多集成逻辑门带有缓冲输出，如集电极开路(OC)或漏极开路(OD)输出。典型的集成逻辑门的输出电流大小在微安(μA)数量级或低于毫安(mA)，例如，标准的 TTL(晶体管－晶体管逻辑电路)的输出电流可以达到 16 mA。大多数发光二极管所需要的电流在 10 mA ~ 50 mA。

例 3.18 对于图 3.40 中执行非－与运算的四输入或非门，确定与输入相对应的输出。

解：当所有的输入都是低电压时，输出才会是高电压，如时序图中的输出波形 X 所示。

相关问题：确定输入 D 反相后的输出，并画出时序图。

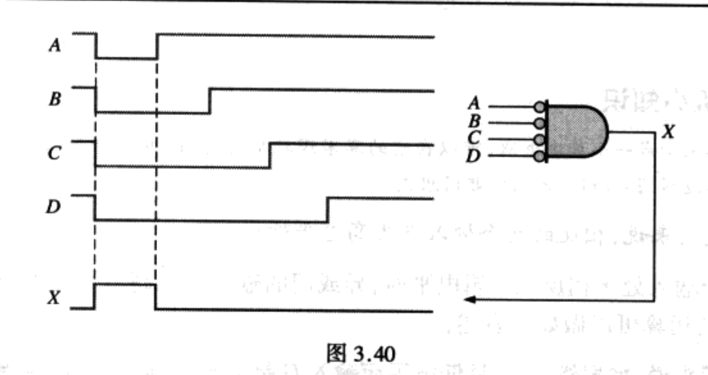


图 3.40

3.5.3 或非门的逻辑表达式

二输入或非门输出的布尔表达式,可以写为

$$X = \overline{A + B}$$

这个等式说明,两个输入变量先进行或运算,然后再求反,求反由或运算表达式上方的横杠指示。运算这个表达式,就可以得到如表 3.10 所示的结果。该或运算表达式可以扩展到多于两个的输入变量,用另外的字母表示其他的变量就可以了。

表 3.10

A	B	$\overline{A+B}=X$
0	0	0 + 0 = 0 = 1
0	1	0 + 1 = 1 = 0
1	0	1 + 0 = 1 = 0
1	1	1 + 1 = 1 = 0

3.6 异或门和同或门

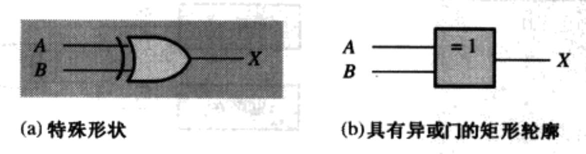
异或门和同或门由一些前面已经讲述过的门组合而成,这将在第 5 章中看到。由于它们在许多应用中起着重要的作用,因此这些门常常作为具有自己特定符号的基本逻辑元件。

学完本节之后,应当能够

- 通过特殊形状符号和矩形轮廓符号来识别异或门和同或门
- 描述异或门和同或门的运算
- 写出异或门和同或门的真值表
- 为具有任意特定输入波形的异或门或同或门画出时序图
- 讨论异或门和同或门应用的例子

3.6.1 异或门

异或门(简写为 XOR)的标准符号如图 3.41 所示。异或门只有两个输入。



(a) 特殊形状　　(b)具有异或门的矩形轮廓

图 3.41 异或门的标准逻辑符号



计算机小知识

异或门连接起来构成一个加法电路,可以在它的算术逻辑单元(ALU)中执行加法、减法、乘法及除法运算。异或门由基本逻辑门 - 与门、或门和非门组成。

◇ 对于异或门来说,相反的两个输入产生高电平输出。

只有当两个输入处于相反的逻辑电平时,异或门的输出才是高电平。根据输入 *A* 和输入 *B* 及输出 *X*,它的运算可以做如下表述:

对于异或门来说,如果输入 *A* 是低电平而输入 *B* 是高电平,或者输入 *A* 是高电平而输出 *B* 是低电平,那么输出 *X* 就是高电平;如果 *A* 和 *B* 都是高电平或者都是低电平,那么输出 *X* 为低电平。

异或门的所有四个可能的输入组合及输出结果,如图 3.42 所示。高电平是有效或者肯定电平,只有当两个输入相反时才会输出高电平。异或门的运算总结在表 3.11。

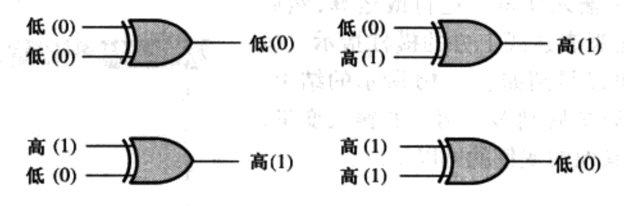


图 3.42 异或门的所有输入逻辑电平和相应的输出电平

例 3.19 一个系统包含两个并行运行的相同电路。只要它们正确运行,两个电路的输出就总是相同的。如其中一个电路没有正确运行,那么同时就会有相反的输出电平。设计一个方法用以检测这两个电路中的一个出现了故障。

解:如图 3.43 所示,电路的输出连接异或门的输入。任何一个电路出现故障都会产生不同的输出,使得异或门的两个输入为相反的电平。这样就会在异或门的输出产生一个高电平,以指示其中的一个电路出现了故障。

相关问题:异或门总是检测图 3.43 中两个电路同时发生故障吗?如果不是,应当处于什么条件。

表 3.11 异或门的真值表

输入		输出
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

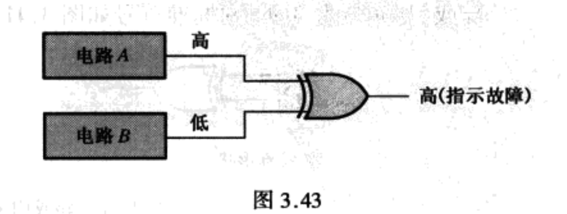
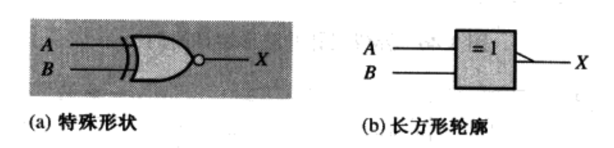


图 3.43

3.6.2 同或门

同或门(简写为XNOR)的标准符号如图3.44所示。和异或门相似,同或门也只有两个输入。同或门符号输出位置上的小圆圈指示它的输出和异或门的输出是相反的。当两个输入逻辑电平相反时,同或门的输出就是低电平。这个运算可以做如下表述(A 和 B 是输入,X 是输出):

对于同或门来说,如果输入 A 是低电平而输入 B 是高电平,或者 A 是高电平,B 是低电平,输出 X 就是低电平;如果输入 A 和输入 B 都是高电平或者都是低电平,输出 X 就是高电平。



(a) 特殊形状　　(b) 长方形轮廓

图3.44 同或门的标准逻辑符号

同或门的4个可能输入组合及输出结果如图3.45所示。同或门的运算总结在表3.12中。注意当两个输入具有相同的电平时,输出就是高电平。

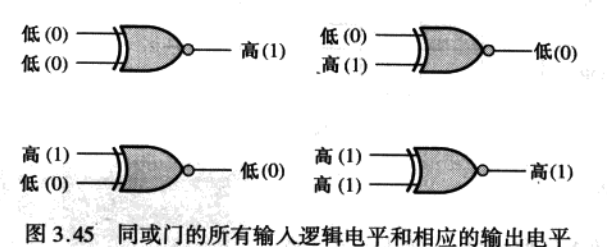
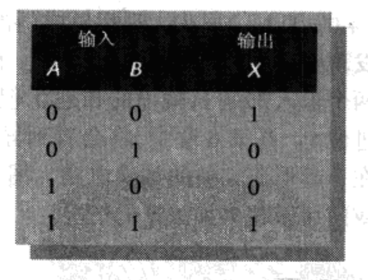


图3.45 同或门的所有输入逻辑电平和相应的输出电平

3.6.3 波形输入运算

表3.12 同或门



输入		输出
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

正如对其他门所做的运算一样,观察异或门和同或门在脉冲输入条件下的运算。和以前一样,在脉冲的每个特定时间段中,对如图3.46所示的异或门应用真值表运算。可以看到在时间段 t_2 和 t_4 期间,输入波形 A 和 B 为相反的电平。所以在这两个时间段,输出 X 是高电平。由于在时间段 t_1 和 t_3 期间,两个输入为相同电平,要么都是高电平要么都是低电平,所以这两个时间段的输出 X 是低电平,如时序图所示。

例3.20 在图3.47中,给定了输入波形 A 和 B,确定异或门和同或门的输出波形。

解: 输出波形如图3.47所示。注意只有当异或门的两个输入相反时,输出才是高电平。只有当同或门的两个输入相同时,输出才是高电平。

相关问题: 如果把以上两个输入波形 A 和 B 反相,请确定输出波形。

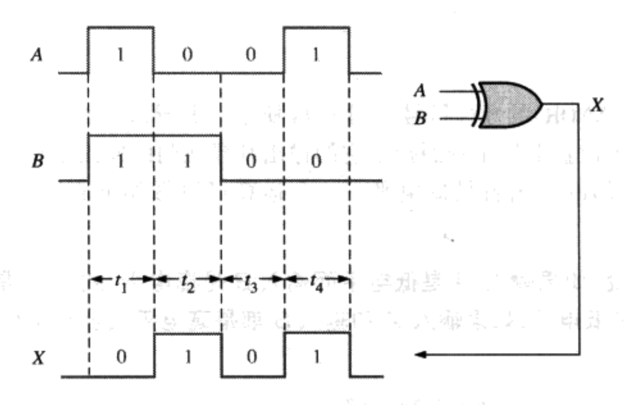


图 3.46 异或门脉冲波形运算的例子

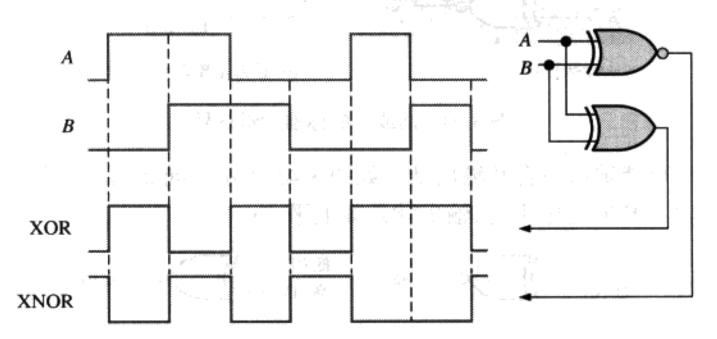
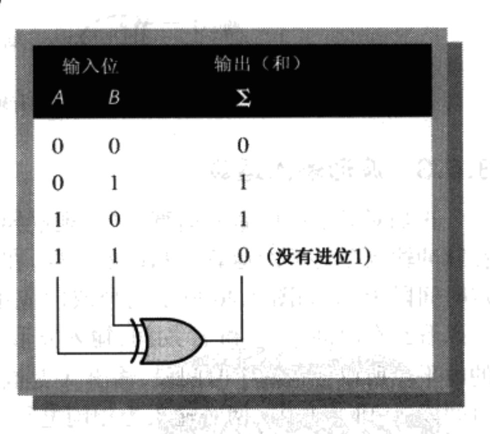


图 3.47

3.6.4 应用举例

异或门可以用做 2 位加法器。记得在第 2 章中,二进制加法的基本规则是:0+0=0,0+1=1,1+0=1,1+1=10。观察异或门的真值表,就会发现输出是两个输入的和运算。在这种情况下,两个输入都是 1,输出的和是 0,但是我们丢弃了进位 1。在第 6 章中,将会看到异或门怎样组合在一起形成完整的加法电路。图 3.48 给出了异或门用做基本加法器的情况。



输入位		输出（和）
A	B	Σ
0	0	0
0	1	1
1	0	1
1	1	0 (没有进位1)

图 3.48 异或门用做两位加法

自测题 (答案在本章的结尾)

1. 当反相器的输入为高电平 1 时,输出是
   (a)高电平或者 1 (b)低电平或者 1 (c)高电平或者 0 (d)低电平或者 0
2. 反相器执行的运算称为
   (a)求反码 (b)确定 (c)反相 (d)答案(a)和(c)
3. 与门的输入为 A、B、C,输出何时为 1(高电平)。
   (a)$A=1,B=1,C=1$ (b)$A=1,B=0,C=1$ (c)$A=0,B=0,C=0$

4. 或门的输入为 A、B、C，输出何时为 1(高电平)。
(a)$A=1,B=1,C=1$　(b)$A=0,B=0,C=1$　(c)$A=0,B=0,C=0$
(d)答案(a)、(b)和(c)　(e)只有答案(a)和(b)

5. 一个脉冲加在一个二输入与非门的每一个输入上。某一个脉冲在 $t=0$ 为高电平，而在 $t=1$ ms 时变为低电平。另一个脉冲在 $t=0.8$ ms 时变为高电平，而在 $t=3$ ms 时变为低电平。输出脉冲应当描述为
(a)在 $t=0$ 时变为低电压，而在 $t=3$ ms 时变为高电平。
(b)在 $t=0.8$ ms 时变为低电平，而在 $t=3$ ms 时变为高电平。
(c)在 $t=0.8$ ms 时变为低电平，而在 $t=1$ ms 时变为高电平。
(d)在 $t=0.8$ ms 时变为低电平，而在 $t=1$ ms 时变为低电平。

6. 一个脉冲应用于一个二输入或非门的每一个输入上。某一个脉冲在 $t=0$ 时为高电平，而在 $t=1$ ms 时变为低电平。另一个脉冲在 $t=0.8$ ms 时变为高电平，而在 $t=3$ ms 时变为低电平。输出脉冲应当描述为：
(a)在 $t=0$ 时变为低电平，而在 $t=3$ ms 时变为高电平。
(b)在 $t=0.8$ ms 时变为低电平，而在 $t=3$ ms 时变为高电平。
(c)在 $t=0.8$ ms 时变为低电平，而在 $t=1$ ms 时变为高电平。
(d)在 $t=0.8$ ms 时变为高电平，而在 $t=1$ ms 时变为低电平。

7. 脉冲加在一个异或门的每一个输入上。某一个脉冲在 $t=0$ 时变为电平，而在 $t=1$ ms 时变为低电平。另一个脉冲在 $t=0.8$ ms 时变为高电平，而在 $t=3$ ms 时变为低电平。输出脉冲应当描述为
(a)在 $t=0$ 时变为高电平，而在 $t=3$ ms 时变为低电平。
(b)在 $t=0$ 时变为高电平，而在 $t=0.8$ ms 时变为低电平。
(c)在 $t=1$ ms 时变为高电平，而在 $t=3$ ms 时变为低电平。
(d)答案(b)和(c)。

8. 一个正向脉冲应用于反相器。从输入前沿到输出前沿的时间间隔为 7 ns。这个参数是
(a)速度功率乘积　(b)传播延迟 t_{PHL}　(c)传播延迟 t_{PLH}　(d)脉冲宽度

习题

3.1 节　反相器

1. 如图 3.49 所示的输入波形加在一个反相器上。画出与输入波形相关的输出波形的时序图。

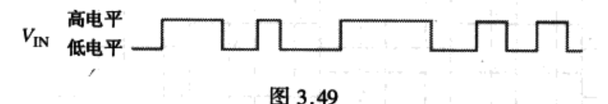


图 3.49

2. 反相器级联网络如图 3.50 所示。如果高电平加在输入点 A，确定点 B 到 F 所对应的输出波形。

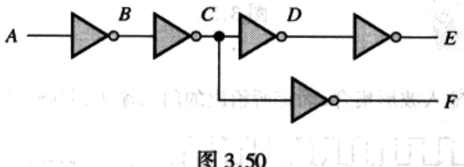


图 3.50

3.2 节　与门

3. 输入波形如图 3.51 所示，确定二输入与门的输出 X，用时序画出与输入所对应的正确的输入输出关系。
4. 如图 3.52，重复习题 3，画出输出波形。

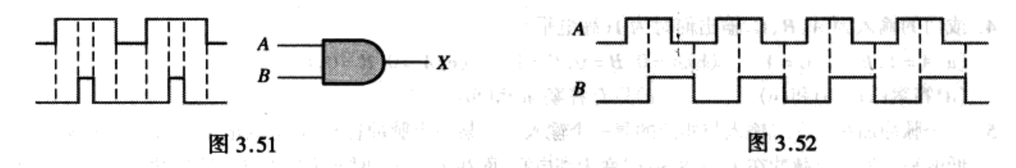


图 3.51　　图 3.52

5. 加在一个三输入与门的输入波形如图 3.53 所示。利用时序图画出与输入对应的输出波形。

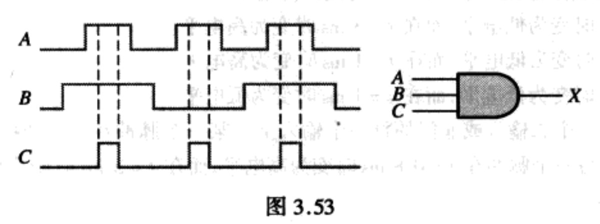


图 3.53

6. 加在一个四输入与门的输入波形如图 3.54 所示。利用时序图画出与输入对应的输出波形。

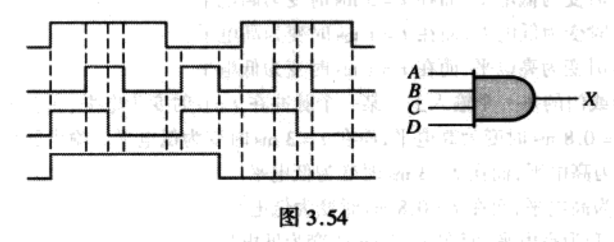


图 3.54

3.3 节　或门

7. 当输入波形如图 3.52 所示时,确定一个二输入或门的输出,并且绘制出时序图。

8. 对三输入或门重复习题 5。

9. 对四输入或门重复习题 6。

10. 对如图 3.55 所示的 5 个输入波形,确定五输入与门的输出,以及五输入或门的输出。画出时序图。

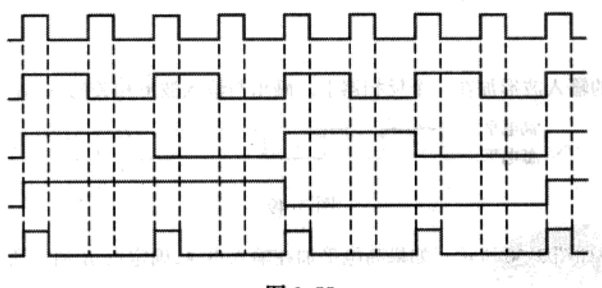


图 3.55

3.4 节　与非门

11. 对于如图 3.56 所示的输入波形集合,确定所给出的门的输出,并画出时序图。

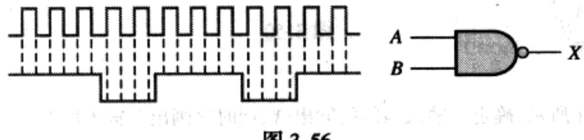


图 3.56

12. 对于如图 3.57 所示的输入波形，确定所给出的门的输出，并画出时序图。

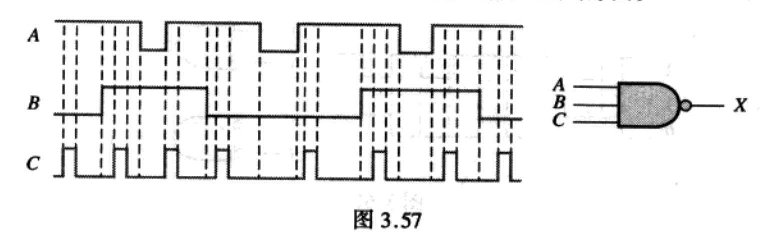


图 3.57

13. 确定如图 3.58 中的输出波形。

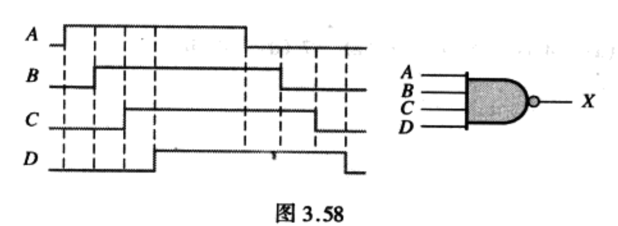


图 3.58

14. 正如所学到的那样，如图 3.59 所示的两个逻辑符号表示等价的运算。两种符号的严格区别在于功能。对于与非门符号来说，两个输入上的高电平给出一个低电平。对于非－或符号来说，输入只要有一个低电平就给出一个高电平输出。利用这两种功能，为给定的输入画出每个门的输出，这两个输出是相同的。

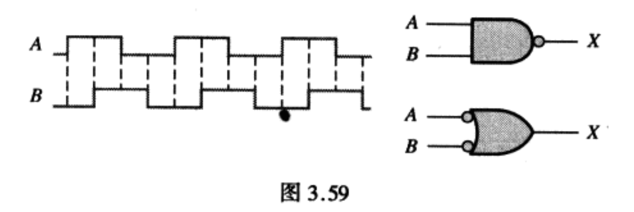


图 3.59

3.5 节 或非门

15. 为一个二输入或非门重复习题 11。
16. 确定图 3.60 中的输出波形，并绘制时序图。

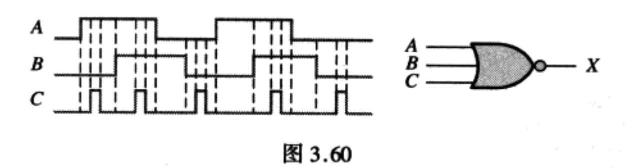


图 3.60

17. 为一个四输入或非门重复习题 13。
18. 与非和非－或符号表示等价运算，但是它们在功能上是不同的。对于或非符号来说，输入只要有一个高电平，就给出低电平输出。对于非－与符号来说，输入上两个低电平才给出一个高电平输出。利用这两种功能，在图 3.61 中对于两个门所给定的输入画出输出波形，这两个输出是相同的。

3.6 节 异或门和同或门

19. 异或门和同或门在逻辑运算上是怎样相互区分的？
20. 为异或门重复习题 11。
21. 为同或门重复习题 11。

22. 为图 3.61 中所示的输入,确定异或门的输出,并画出时序图。

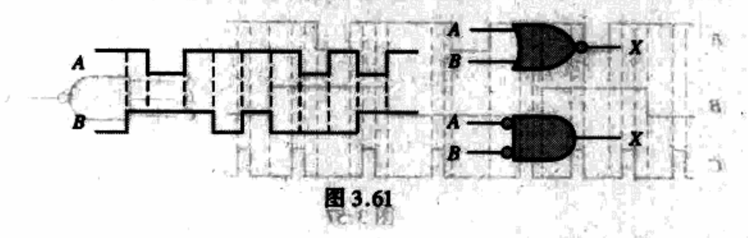
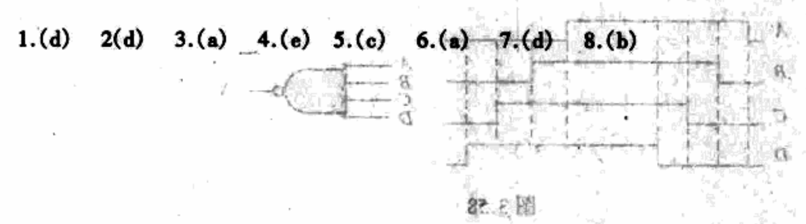
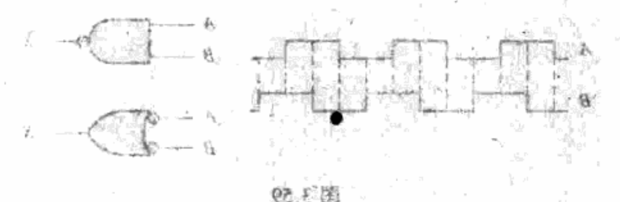


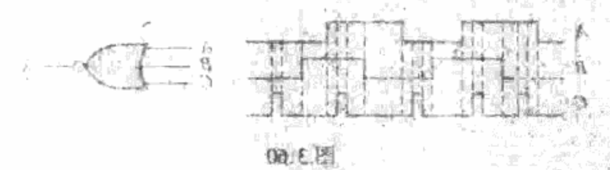
图 3.61

自测题答案

1.(d)　2(d)　3.(a)　4.(e)　5.(c)　6.(a)　7.(d)　8.(b)







第 4 章　布尔代数和逻辑化简

章节提纲

4.1　布尔运算和表达式

布尔代数是数字系统的数学。布尔代数的基本知识对于学习和分析逻辑电路是必不可少的。上一章介绍了非、与、或、与非和或非门的布尔运算和表达式。

学完本节以后,应当能够

- 定义变量
- 定义文字
- 识别和项
- 计算和项
- 识别乘积项
- 计算乘积项
- 解释布尔加法
- 解释布尔乘法

布尔代数中使用的术语为变量、反码和文字。变量是用来表示逻辑数量的符号(通常是斜体大写字母)。任何一个单变量可以具有 1 或者 0 的值。反码是变量的反相,并且由变量上方的横杠(上划杠)表示。例如,变量 A 的反码是 $\bar{A}$。如果 $A=1$,那么 $\bar{A}=0$;如果 $A=0$,那么 $\bar{A}=1$。变量 A 的反码读做"非 A"或者"A 横杠"。有时候用撇符号而不是用上划杠来表示变量的反码;例如,B' 就表示 B 的反码。在本书中,使用的是上划杠。一个文字就是一个变量或者变量的反码。



计算机小知识

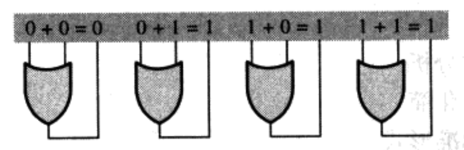
在微处理器中,算术逻辑单元(ALU)根据程序的指令,对数字数据执算术和布尔逻辑运算。逻辑运算等

价于熟知的基本逻辑门的运算,但是每次至少处理8位。布尔逻辑指令的例子是与、或、非和异或,它们被称为助记符。汇编语言程序使用助记符来指定运算。另一个称为汇编器的程序把助记符翻译成可以被微处理器理解的二进制代码。

4.1.1 布尔加法

◇ 或门就是一个布尔加。

回顾第3章,布尔加等价于或运算,其基本法则用或门表示如下:



在布尔代数中,和项就是文字变量的相加。在逻辑电路中,和项由或运算产生,其中没有与运算。和项的一些例子是:$A+B$, $A+\overline{B}$, $A+B+\overline{C}$, $\overline{A}+B+C+\overline{D}$。

和项中有一个或者多个文字为1时,和项就等于1。只有当每个文字都是0时,和项才等于0。

例4.1 已知 $A+\overline{B}+C+\overline{D}=0$,求出 A、B、C 和 D 的值。

解: 要使得和项等于0,那么和项中的每一个文字必须为0。因此 $A=\mathbf{0}$, $\overline{B}=0$ 那么 $B=\mathbf{1}$, $C=\mathbf{0}$, $\overline{D}=0$ 那么 $D=\mathbf{1}$。

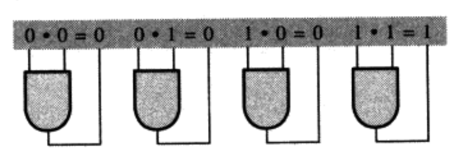
$$A+\overline{B}+C+\overline{D}=0+\overline{1}+0+\overline{1}=0+0+0+0=0$$

相关问题: 已知 $\overline{A}+B=0$,求出 A、B 的值。

4.1.2 布尔乘法

◇ 与门就是一个布尔乘法器。

同样回顾第3章,布尔乘法等价于与门运算,其基本法则用与门表示如下:



在布尔代数中,乘积项就是文字的乘积。在逻辑电路中,乘积项由与门运算产生,没有或的运算。乘积项的一些例子为:AB, $A\overline{B}$, ABC, $A\overline{B}C\overline{D}$。

当一个乘积项的每一个文字都是1时,乘积项等于1。当一个或者多个文字为0时,乘积项就等于0。

例4.2 确定使得乘积项 $A\overline{B}C\overline{D}$ 等于1的 A、B、C 和 D 的数值。

解: 为了使得乘积项等于1,那么该项中的每一个文字都必须是1。所以,$A=\mathbf{1}$, $\overline{B}=1$ 那么 $B=\mathbf{0}$, $C=\mathbf{1}$, $\overline{D}=1$ 那么 $D=\mathbf{0}$。

$$A\overline{B}C\overline{D}=1\cdot\overline{0}\cdot1\cdot\overline{0}=1\cdot1\cdot1\cdot1=1$$

相关问题: 确定使得乘积项 $\overline{A}\overline{B}$ 等于1的 A 和 B 的值。

4.2　布尔代数的定理和法则

和其他数学领域一样,要正确使用布尔代数,必须遵循一些严格推导出来的法则和定理。其中一些最重要的概念将在本节介绍。

学完本节之后,应当能够

- 应用加法和乘法的交换律
- 应用加法和乘法的结合律
- 应用分配律
- 应用布尔代数的 12 个基本法则

4.2.1　布尔代数的定律

布尔代数的基本定律——加法和乘法的交换律、加法和乘法的结合律及分配律——和通常的代数的定律一样。每一个定律都会由两个或者三个变量来表述,但是变量的个数不局限于此。

交换律　两个变量加法的交换律可以写为

$$A + B = B + A \tag{4.1}$$

这个定律表明对变量进行或运算的次序不会对结果产生影响。记住,布尔代数应用于逻辑电路时,加法和或运算是一样的。图 4.1 说明了应用于或门的交换律,并且指出变量加在哪个输入上并不重要。(符号≡的意思是"恒等于"。)

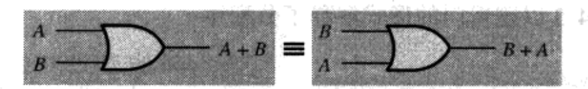


图 4.1　加法交换律的应用

两个变量乘法的交换律为

$$AB = BA \tag{4.2}$$

这个定律表明变量与运算的次序不会对结果产生影响。图 4.2 给出了这个定律应用在与门的情况。

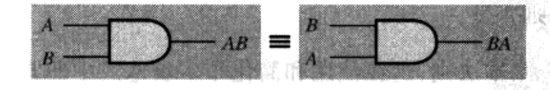


图 4.2　乘法交换律的应用

结合律　3 个变量的加法结合律可以写为下面的形式:

$$A + (B + C) = (A + B) + C \tag{4.3}$$

这个定律表明在对多于两个的变量进行或运算时,无论把哪些变量分为一组,其结果相同。图 4.3给出了这个定律应用于二输入或门的情况。

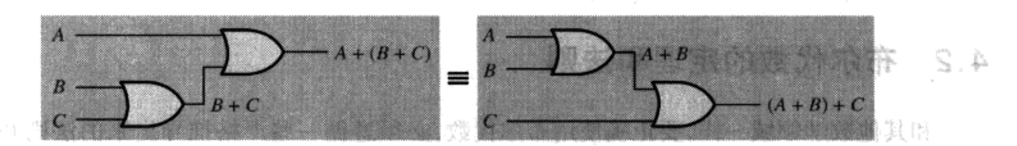


图 4.3 加法结合律的应用

3 个变量的乘法的结合律可以写为

$$A(BC) = (AB)C \tag{4.4}$$

这个定律表明当对多于两个的变量进行与运算时,变量被分组的次序不会对结果产生影响。图 4.4 给出了这个定律应用于二输入与门的情况。

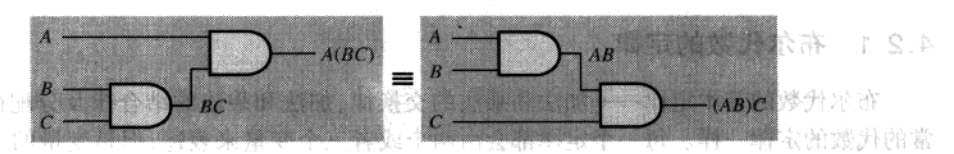


图 4.4 乘法交换律的应用

分配律 3 个变量的分配律可以写为

$$A(B + C) = AB + AC \tag{4.5}$$

这个定律表明对两个或者更多的变量进行或运算,然后把结果和一个单变量相与,等价于把这个单变量和这两个或者更多变量中的每一个变量分别进行与运算,然后再把这些乘积进行或运算。分配律同样表示了因子分解的过程,共同变量 A 从来乘积项中分解出来,例如,$AB + AC = A(B + C)$。图 4.5 以逻辑门的形式解释了分配律。

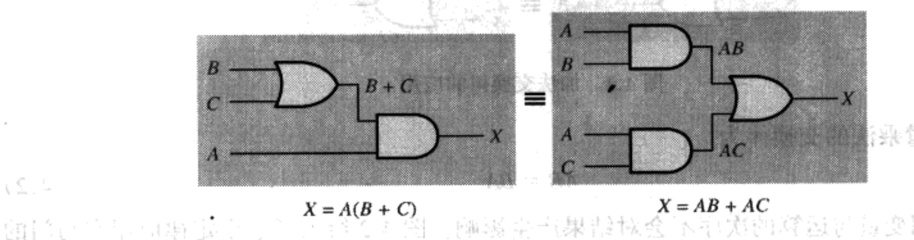


图 4.5 分配律的应用

4.2.2 布尔代数法则

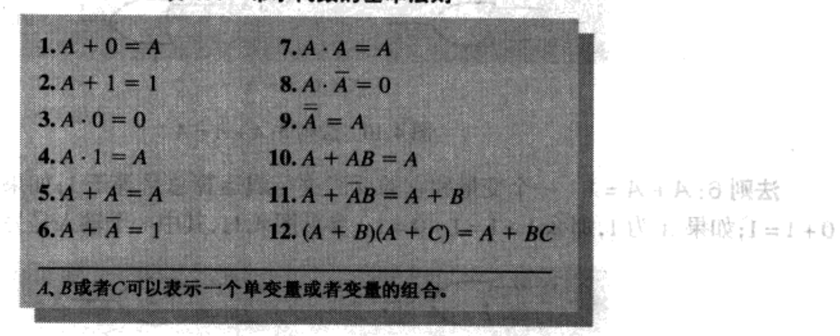
表 4.1 列出了 12 个基本法则,这在使用和简化布尔表达式中非常有用。法则 1 到 9 以它们在逻辑门中应用的方法得到介绍。法则 10 到 12 将会从先前已讨论的简单定律和法则中推导出来。

法则 1: $A + 0 = A$ 一个变量和 0 进行或运算总是等于变量本身。如果输入变量是 1,输出变量 X 就是 1,也就是等于 A。如果 A 是 0,那么输出就是 0,也等于 A。这个法则如图 4.6 所示,其中第二个输入被固定为 0。

法则 2: $A + 1 = 1$ 一个变量和 1 进行或运算总是等于 1。或门的一个输入为 1,无论另一

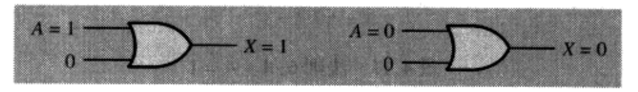
个输入的变量值是多少，所产生的输出就会是 1。这个法则如图 4.7 所示，其中第二个输入固定为 1。

表 4.1 布尔代数的基本法则



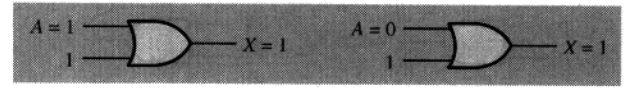
1. $A+0=A$	7. $A\cdot A=A$
2. $A+1=1$	8. $A\cdot\overline{A}=0$
3. $A\cdot 0=0$	9. $\overline{\overline{A}}=A$
4. $A\cdot 1=A$	10. $A+AB=A$
5. $A+A=A$	11. $A+\overline{A}B=A+B$
6. $A+\overline{A}=1$	12. $(A+B)(A+C)=A+BC$

A、*B*或者*C*可以表示一个单变量或者变量的组合。



$X=A+0=A$

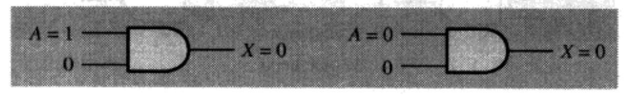
图 4.6 法则 1：$A+0=A$



$X=A+1=1$

图 4.7 法则 2：$A+1=1$

法则 3：$A\cdot 0=0$ 一个变量和 0 进行与运算总是等于 0。任何时候与门的一个输入为 0，无论另一个输入上的变量值是多少，输出就是 0。这个法则如图 4.8 所示，其中第二个输入固定为 0。



$X=A\cdot 0=0$

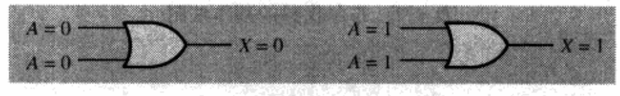
图 4.8 法则 3：$A\cdot 0=0$

法则 4：$A\cdot 1=A$ 一个变量和 1 进行与运算总是等于变量本身。如果 A 为 0，与门的输出就是 0；如果 A 为 1，与门的输出就是 1，因为现在这两个输入都是 1。这个法则如图 4.9 所示，其中第二个输入固定为 1。

$X=A\cdot 1=A$

图 4.9 法则 4：$A\cdot 1=A$

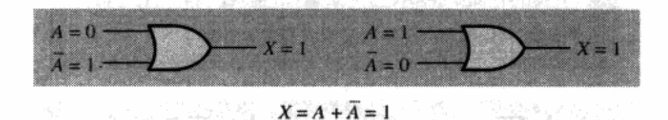
法则 5: $A+A=A$　一个变量和它本身进行或运算总是等于变量本身。如果 A 为 0,那么 $0+0=0$;如果 A 为 1,那么 $1+1=1$。这个法则如图 4.10 所示,其中两个输入都是相同的变量。



$X=A+A=A$

图 4.10　法则 5: $A+A=A$

法则 6: $A+\overline{A}=1$　一个变量和它的反码进行或运算总是等于 1;如果 A 为 0,那么 $0+\overline{0}=0+1=1$;如果 A 为 1,那么 $1+\overline{1}=1+0=1$。参见图 4.11,其中一个输入是另一个输入的反码。



$X=A+\overline{A}=1$

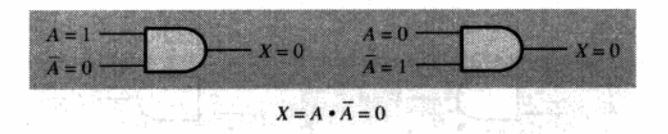
图 4.11　法则 6: $A+\overline{A}=1$

法则 7: $A\cdot A=A$　一个变量和它本身进行与运算总是等于变量本身。如果 $A=0$,那么 $0\cdot 0=0$;如果 $A=1$,那么 $1\cdot 1=1$。图 4.12 解释了这个法则。

$X=A\cdot A=A$

图 4.12　法则 7: $A\cdot A=A$

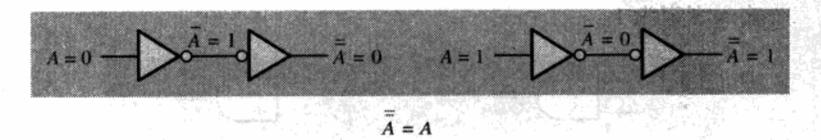
法则 8: $A\cdot\overline{A}=0$　一个变量和它的反码进行与运算总是等于 0。A 或者 $\overline{A}$ 总有一个为 0;当 0 应用于与门的输入时,输出就是 0。图 4.13 解释了这个法则。



$X=A\cdot\overline{A}=0$

图 4.13　法则 8: $A\cdot\overline{A}=0$

法则 9: $\overline{\overline{A}}=A$　对一个变量进行两次求反后总是等于变量本身。如果开始于变量 A,对其进行一次求反(反相),你就得到 $\overline{A}$。如果再对 $\overline{A}$ 进行求反就会得到 A,也就是原始变量。这个法则如图 4.14 所示,图中使用了反相器。



$\overline{\overline{A}}=A$

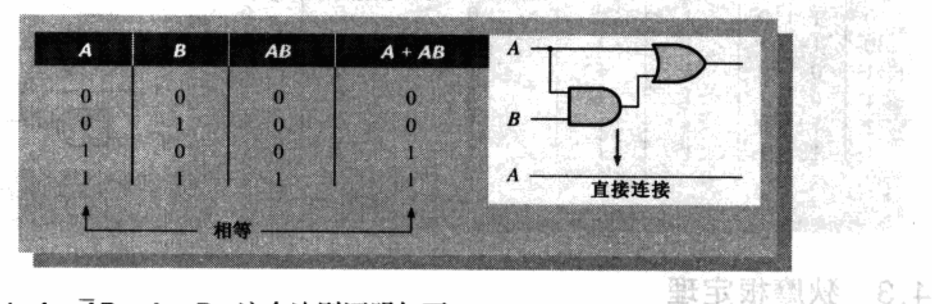
图 4.14　法则 9: $\overline{\overline{A}}=A$

法则 10: $A+AB=A$　这个法则可以使用分配律(法则 2 和法则 4)来证明,如下所示:

$$
\begin{aligned}
A + AB &= A(1 + B) && \text{因式分解(分配律)} \\
&= A \cdot 1 && \text{法则 2:}(1 + B) = 1 \\
&= A && \text{法则 4: } A \cdot 1 = A
\end{aligned}
$$

这个证明如表 4.2 所示,其中给出了真值表和简化的逻辑电路。

表 4.2 法则 10: $A + AB = A$



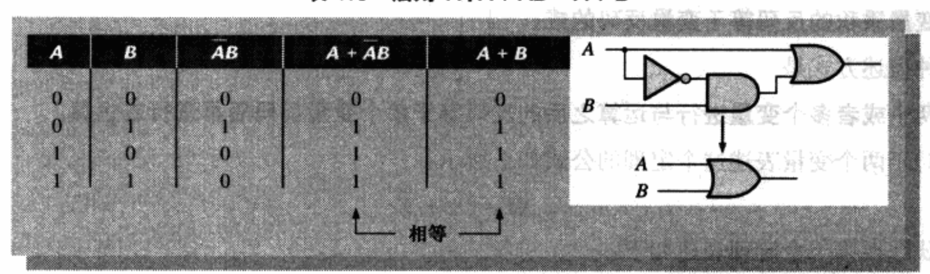
A	B	AB	$A+AB$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

法则 11: $A + \overline{A}B = A + B$ 这个法则证明如下:

$$
\begin{aligned}
A + \overline{A}B &= (A + AB) + \overline{A}B && \text{法则 10:} A = A + AB \\
&= (AA + AB) + \overline{A}B && \text{法则 7:} A = AA \\
&= AA + AB + A\overline{A} + \overline{A}B && \text{法则 8:加 } A\overline{A} = 0 \\
&= (A + \overline{A})(A + B) && \text{因式分解} \\
&= 1 \cdot (A + B) && \text{法则 6:} A + \overline{A} = 1 \\
&= A + B && \text{法则 4: 舍去 1}
\end{aligned}
$$

表 4.3 给出了真值表和简化的逻辑电路。

表 4.3 法则 11: $A + \overline{A}B = A + B$



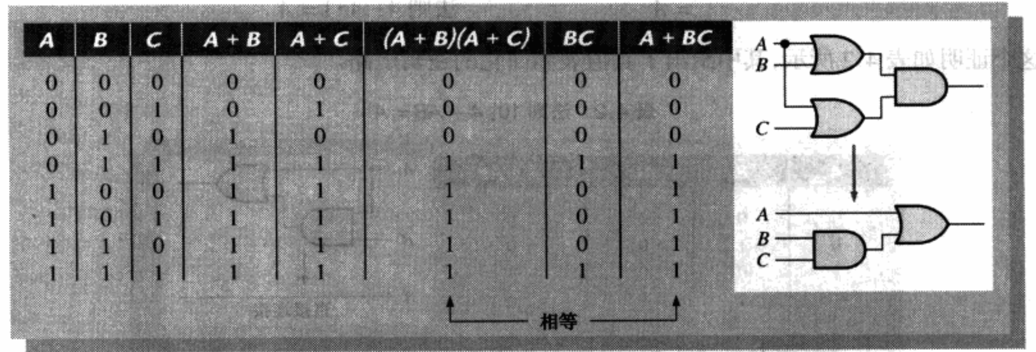
A	B	$\overline{A}B$	$A+\overline{A}B$	$A+B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

法则 12: $(A + B)(A + C) = A + BC$ 这个法则的证明如下所示:

$$
\begin{aligned}
(A + B)(A + C) &= AA + AC + AB + BC && \text{分配律} \\
&= A + AC + AB + BC && \text{法则 7:} AA = A \\
&= A(1 + C) + AB + BC && \text{因式分解(分配律)} \\
&= A \cdot 1 + AB + BC && \text{法则 2:} 1 + C = 1 \\
&= A(1 + B) + BC && \text{因式分解(分配律)} \\
&= A \cdot 1 + BC && \text{法则 2:} 1 + B = 1 \\
&= A + BC && \text{法则 4:} A \cdot 1 = A
\end{aligned}
$$

这个证明如表 4.4 所示,其中给出了真值表和简化的逻辑电路。

表 4.4 法则 12:$(A+B)(A+C)=A+BC$



A	B	C	$A+B$	$A+C$	$(A+B)(A+C)$	BC	$A+BC$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

4.3 狄摩根定理

数学家狄摩根(DeMorgan)提出的两个定理,成为了布尔代数中的重要一部分。在实际术语中,狄摩根定理提供了与非门和非-或门及或非门和非-与门之间等价的数学证明,这在第 3 章讨论过。

学完本节之后,应当能够

- 表述狄摩根定理
- 把狄摩根定理和与非门、非-或门之间的等价及或非门和非-与门之间进行关联
- 把狄摩根定理应用于布尔表达式的简化

狄摩根的一个定理如下所述:

变量乘积的反码等于变量反码的或。

另一种表述方式是

两个或者多个变量进行与运算之后的反码等于单个变量反码后再进行或运算。

对于两个变量表述这个定理的公式如下所示:

$$\overline{XY}=\overline{X}+\overline{Y} \tag{4.6}$$

狄摩根第二个定理表述如下:

变量之和的反码等于变量反码的乘积。

另一种表述方式是

对两个以上变量进行或运算之后的反码等于单个变量反码再进行与运算的结果。

对于两个变量,表述这个定理的公式为

$$\overline{X+Y}=\overline{X}\,\overline{Y} \tag{4.7}$$

图 4.15 给出了式(4.6)和式(4.7)的真值表和等价的逻辑门。

如表述的那样,狄摩根定理也可应用于多于两个变量的表达式中。下面的例子就解释了狄摩根定理应用到 3 变量和 4 变量表达式的情况。

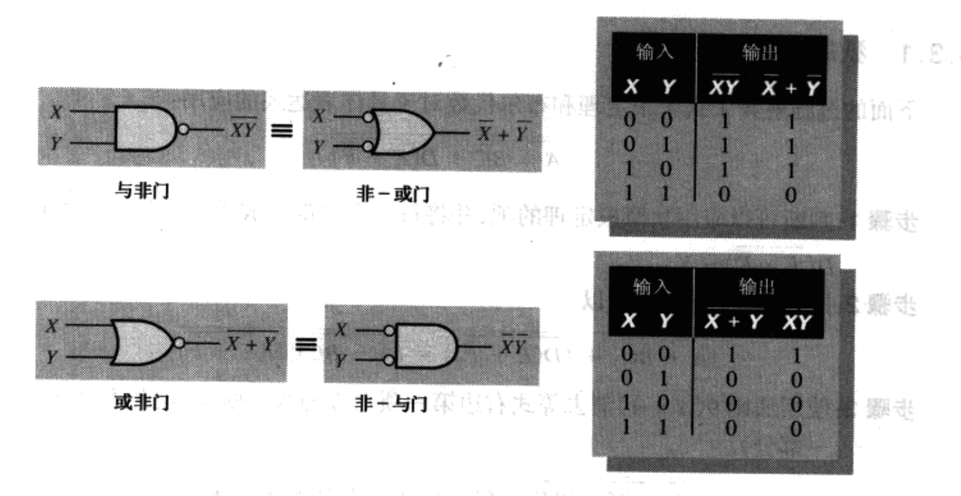


图 4.15 用于解释狄摩根定理的等价门和相应的真值表。注意每个表中的两个相同的输出列。这表示了等价门执行相同的逻辑运算

例 4.3 摩根定理应用于表达式$\overline{XYZ}$和$\overline{X+Y+Z}$。

解：

$$\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$$
$$\overline{X+Y+Z} = \overline{X}\,\overline{Y}\,\overline{Z}$$

相关问题：摩根定理应用于表达式$\overline{\overline{X}+\overline{Y}+\overline{Z}}$。

例 4.4 把摩根定理应用于表达式$\overline{WXYZ}$和$\overline{W+X+Y+Z}$。

解：

$$\overline{WXYZ} = \overline{W} + \overline{X} + \overline{Y} + \overline{Z}$$
$$\overline{W+X+Y+Z} = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$$

相关问题：将狄摩根定理应用于表达式$\overline{\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}}$。

狄摩根定理中的每一个变量，如式(4.6)和式(4.7)所示，也可以表示其他变量的组合。例如，X 可以等于 $AB+C$ 项，而 Y 可以等于 $A+BC$ 项。所以如果可以把两个变量的狄摩根定理表示$\overline{XY}=\overline{X}+\overline{Y}$ 应用于表达式$\overline{(AB+C)(A+BC)}$，就会得到以下结果：

$$\overline{(AB+C)(A+BC)} = (\overline{AB+C}) + (\overline{A+BC})$$

注意前面的结果有两个项：$\overline{AB+C}$和$\overline{A+BC}$，那么可以对上述两项再次使用狄摩根定理$\overline{X+Y}=\overline{X}\,\overline{Y}$，如下所示：

$$(\overline{AB+C}) + (\overline{A+BC}) = (\overline{AB})\overline{C} + \overline{A}(\overline{BC})$$

注意表达式中具有两项，可以再一次使用狄摩根定理。这两个项是$\overline{AB}$和$\overline{BC}$。最后一次应用狄摩根定理给出了如下的结果：

$$(\overline{AB})\overline{C} + \overline{A}(\overline{BC}) = (\overline{A}+\overline{B})\overline{C} + \overline{A}(\overline{B}+\overline{C})$$

虽然可以使用布尔法则和定律进一步进行简化，但是已经不能再使用狄摩根定理了。

4.3.1 狄摩根定理的应用

下面的过程解释了狄摩根定理和布尔代数对于具体表达式的应用：

$$\overline{\overline{A + B\overline{C}} + D(\overline{E + \overline{F}})}$$

步骤 1: 判断可以应用狄摩根定理的项，并将每一个项都看成单个变量。让 $\overline{A + B\overline{C}} = X$ 和 $D(\overline{E + \overline{F}}) = Y$。

步骤 2: 因为 $\overline{X + Y} = \overline{X}\,\overline{Y}$，所以

$$\overline{(\overline{A + B\overline{C}}) + (D(\overline{E + \overline{F}}))} = (\overline{\overline{A + B\overline{C}}})(\overline{D(\overline{E + \overline{F}})})$$

步骤 3: 使用法则 9($\overline{\overline{A}} = A$)消去等式右边第一项上方的双重横杠(这并不是狄摩根定理的一部分)。

$$(\overline{\overline{A + B\overline{C}}})(\overline{D(\overline{E + \overline{F}})}) = (A + B\overline{C})(\overline{D(\overline{E + \overline{F}})})$$

步骤 4: 为等式右边第二项应用狄摩根定理。

$$(A + B\overline{C})(\overline{D(\overline{E + \overline{F}})}) = (A + B\overline{C})(\overline{D} + (\overline{\overline{E + \overline{F}}}))$$

步骤 5: 使用法则 9($\overline{\overline{A}} = A$)消去 $E + \overline{F}$ 部分项上方的双重横杠。

$$(A + B\overline{C})(\overline{D} + \overline{\overline{E + \overline{F}}}) = (A + B\overline{C})(\overline{D} + E + \overline{F})$$

下面的三个例子将进一步解释怎样应用狄摩根定理。

例 4.5 对下面的表达式应用狄摩根定理。

(a) $\overline{(A + B + C)D}$ (b) $\overline{ABC + DEF}$ (c) $\overline{A\overline{B} + \overline{C}D + EF}$

解：

(a) 让 $A + B + C = X$ 和 $D = Y$。表达式 $\overline{(A + B + C)D}$ 就具有了 $\overline{XY} = \overline{X} + \overline{Y}$ 形式，因而可以写成

$$\overline{(A + B + C)D} = \overline{A + B + C} + \overline{D}$$

下一步，为 $\overline{A + B + C}$ 应用狄摩根定理。

$$\overline{A + B + C} + \overline{D} = \overline{A}\,\overline{B}\,\overline{C} + \overline{D}$$

(b) 让 $ABC = X$、$DEF = Y$，表达式 $\overline{ABC + DEF}$ 就具有了 $\overline{XY} = \overline{X + Y}$ 形式，因而可以写成

$$\overline{ABC + DEF} = (\overline{ABC})(\overline{DEF})$$

下一步，为上式右边的项 $\overline{ABC}$ 和 $\overline{DEF}$ 应用狄摩根定理。

$$(\overline{ABC})(\overline{DEF}) = (\overline{A} + \overline{B} + \overline{C})(\overline{D} + \overline{E} + \overline{F})$$

(c) 让 $A\overline{B} = X$、$\overline{C}D = Y$ 和 $EF = Z$。表达式 $\overline{A\overline{B} + \overline{C}D + EF}$ 就具有了 $\overline{X + Y + Z} = \overline{X}\,\overline{Y}\,\overline{Z}$ 形式，因而可以写成

$$\overline{A\overline{B} + \overline{C}D + EF} = (\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF})$$

下一步,为上式右边的项$\overline{A\overline{B}}$、$\overline{\overline{C}D}$和$\overline{EF}$应用狄摩根定理。

$$(\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF}) = (\overline{A} + B)(C + \overline{D})(\overline{E} + \overline{F})$$

相关问题:为表达式$\overline{\overline{ABC} + D + E}$应用狄摩根定理。

例 4.6 对每个表达式使用狄摩根定理:

(a) $\overline{\overline{(A+B)} + \overline{C}}$ (b) $\overline{(\overline{A}+B) + CD}$ (c) $\overline{(A+B)\overline{CD} + E + \overline{F}}$

解:

(a) $\overline{\overline{(A+B)} + \overline{C}} = \overline{\overline{(A+B)}}\,\overline{\overline{C}} = (A+B)C$

(b) $\overline{(\overline{A}+B) + CD} = \overline{(\overline{A}+B)}\,\overline{CD} = (\overline{\overline{A}}\overline{B})(\overline{C}+\overline{D}) = A\overline{B}(\overline{C}+\overline{D})$

(c) $\overline{(A+B)\overline{CD} + E + \overline{F}} = (\overline{(A+B)\overline{CD}})(\overline{E+\overline{F}}) = (\overline{A}\overline{B} + C + D)\overline{E}F$

相关问题:对表达式$\overline{\overline{A}B(C+\overline{D}) + E}$应用狄摩根定理。

例 4.7 异或门(XOR)的布尔表达式为 $A\overline{B} + \overline{A}B$。用此作为开始点,应用狄摩根定理及任何其他可以使用的法则,推导出一个同或门(XNOR)的表达式。

解:从异或门表达式的反相开始,然后应用狄摩根定理,如下所示:

$$\overline{A\overline{B} + \overline{A}B} = (\overline{A\overline{B}})(\overline{\overline{A}B}) = (\overline{A} + \overline{\overline{B}})(\overline{\overline{A}} + \overline{B}) = (\overline{A} + B)(A + \overline{B})$$

下一步,使用分配律和法则 8($A \cdot \overline{A} = 0$):

$$(\overline{A} + B)(A + \overline{B}) = \overline{A}A + \overline{A}\,\overline{B} + AB + B\overline{B} = \overline{A}\,\overline{B} + AB$$

同或门的最终表达式为 $\overline{A}\overline{B} + AB$。注意只要两个变量都为 0 或者都为 1 时,这个表达式就等 1。

相关问题:从 4 输入与非门的表达式开始,使用狄摩根定理推导出一个 4 输入非 - 或门的表达式。

4.4 逻辑电路的布尔分析

布尔代数提供了一种简洁的方式,用以表达由逻辑门组成的逻辑电路的运算,从而可以确定不同输入组合所对应的输出。

学完本节以后,应当能够

- 为逻辑门组合确定布尔表达式
- 通过布尔表达式进行电路的逻辑运算
- 构建真值表

4.4.1 逻辑电路的布尔表达式

◇ 逻辑电路可以用布尔等式来描述。

为了推导出一个给定逻辑电路的布尔表达式,从最左面的输入开始一直到最后的输出,写出每个门的表达式。对于图 4.16 中的示例电路,布尔表达式的确定如下所示:

1. 最左面输入 C 和 D 的与门的表达式为 CD。

2. 最左面与门的输出是或门的一个输入而 B 是另一个输入。所以,或门的表达式为 $B+CD$。
3. 或门的输出是最右边与门的一个输入而 A 是另一个输入。所以,该与门的表达式为 $A(B+CD)$,这就是整个电路最终的表达式。

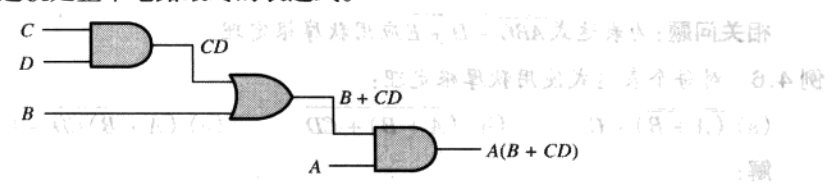


图 4.16 一个逻辑电路,给出了输出布尔表达式的推导过程

4.4.2 构建逻辑电路的真值表

◇ 逻辑电路可以用真值表来描述。

一旦给定逻辑电路的布尔表达式得到确定,就可以构建其真值表,真值表用来展示输入变量的所有可能值所对应的输出。这个过程需要为输入变量所有可能数值的组合来评估布尔表达式。对于图 4.16 所示的电路,有 4 个输入变量(A、B、C 和 D),所以有 16($2^4=16$)种可能的数值组合。

表达式的估算 为了对表达式 $A(B+CD)$ 做出评估,首先使用布尔加法和乘法法则,寻找使得表达式等于 1 的变量的值。在这种情况下,只有当 $A=1$ 及 $B+CD=1$ 的时候,表达式才等于 1,因为

$$A(B+CD)=1\cdot 1=1$$

现在确定什么时候 $B+CD$ 项等于 1。如果 $B=1$ 或者 $CD=1$,或者 B 和 CD 都等于 1,那么 $A(B+CD)$ 项就等于 1,因为

$$B+CD=1+0=1$$
$$B+CD=0+1=1$$
$$B+CD=1+1=1$$

只有 $C=1$ 和 $D=1$ 时,CD 项就等于 1。

总之,当 $A=1$ 及 $B=1$ 而不用管 C 和 D 的数值时,或者当 $A=1$ 及 $C=1$ 并且 $D=1$ 而不用管 B 的数值时,表达式 $A(B+CD)$ 就等于 1。对于变量的所有其他数值组合,表达式 $A(B+CD)=0$。

结果放入真值表 第一步,如表 4.5 所示,按照二进制的序列把输入变量 0 和 1 的 16 种组合列出来。下一步,对于在估算中已经确定表达式为 1 的输入变量的每一种组合,在对应的输出列上写上 1。对于所有其他的输入变量组合,在相应的输出列中写上 0。这些结果如表 4.5。

表 4.5 图 4.16 中逻辑电路的真值表

输入				输出
A	B	C	D	$A(B+CD)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

4.5　用布尔代数进行化简

在布尔代数的应用中，许多时候必须把特定的表达式化简为它的最简单形式或把它的形式变得更加简易，以给出最有效的表达式。本节所采用的方法就是使用布尔代数的基本定律、法则及定理来操作和化简表达式。熟练掌握这些方法取决于对布尔代数透彻理解及大量的应用练习。

学完本节以后，应当能够

■ 应用布尔代数的定律、法则及定理化简一般表达式

简化的布尔表达式使用尽可能少的门，实现给定表达式的功能。例 4.8 到例 4.11 给出了布尔代数的化简。

例 4.8　使用布尔代数简化下面的表达式：

$$AB + A(B + C) + B(B + C)$$

解：下面所给出的步骤不一定是唯一的方法。

步骤 1：为表达式中的第二项和第三项应用分配律，如下所示：

$$AB + AB + AC + BB + BC$$

步骤 2：为等式右边第四项应用法则 7($BB = B$)：

$$AB + AB + AC + B + BC$$

步骤 3：为等式右边第三项应用法则 5($AB + AB = AB$)：

$$AB + AC + B + BC$$

步骤 4：为等式右边最后两项应用法则 10($B + BC = B$)：

$$AB + AC + B$$

步骤 5：为等式右边第一项和第三项应用法则 10($AB + B = B$)：

$$B + AC$$

在这里，表达式已被尽可能地简化了。一旦在应用布尔代数中取得了经验，就可以经常合并许多独立的步骤。

相关问题：化简布尔表达式 $A\bar{B} + A(\overline{B + C}) + B(\overline{B + C})$。

图 4.17 展示了例 4.8 中的化简过程，它大幅度减少了实现该表达式的逻辑门的个数。图 4.17(a)展示了在原来形式下需要 5 个门来实现该表达式；然而简化的表达式只需要两个门，如图 4.17(b)所示。认识到这两个门电路是等价的，这一点是非常重要的。也就是说，A、B、C 输入的任意电平组合，通过这两个电路都得到一样的输出。

◇ 化简意味着用更少的门实现一样的功能。

例 4.9　化简下面的布尔表达式：

$$[A\bar{B}(C + BD) + \bar{A}\bar{B}]C$$

注意:中括号和圆括号表示一样的意思:里面的项乘以(与)外面的项。

解:

步骤1:对上式中括号里面的项应用分配律:

$$(A\overline{B}C + A\overline{B}BD + \overline{A}\,\overline{B})C$$

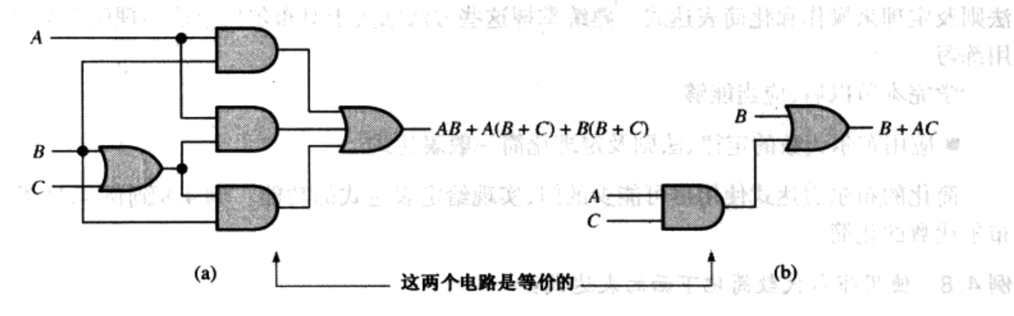


图 4.17 例 4.8 的门电路

步骤2:对上式圆括号里面的第二项应用法则 8($\overline{B}B = 0$):

$$(A\overline{B}C + A \cdot 0 \cdot D + \overline{A}\,\overline{B})C$$

步骤3:对上式圆括号里面的第二项应用法则 3($A \cdot 0 \cdot D = 0$):

$$(A\overline{B}C + 0 + \overline{A}\,\overline{B})C$$

步骤4:对上式圆括号内部应用法则 1(舍去 0):

$$(A\overline{B}C + \overline{A}\,\overline{B})C$$

步骤5:对上式应用分配律:

$$A\overline{B}CC + \overline{A}\,\overline{B}C$$

步骤6:对上式第一项应用法则 7($CC = C$):

$$A\overline{B}C + \overline{A}\,\overline{B}C$$

步骤7:对上式进行因式分解,分解出

$$\overline{B}C(A + \overline{A})$$

步骤8:对上式应用法则 6($A + \overline{A} = 1$):

$$\overline{B}C \cdot 1$$

步骤9:对上式应用法则 4(舍去 1):

$$\overline{B}C$$

相关问题:化简布尔表达式$[AB(C + \overline{BD}) + \overline{AB}]CD$。

例 4.10 化简下面的布尔表达式:

$$\overline{A}BC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + ABC$$

解：

步骤 1：从第一项和最后一项中分解出 BC：

$$BC(\overline{A} + A) + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C$$

步骤 2：对上式圆括号中的项应用法则 6($\overline{A} + A = 1$)，并且从第二项和最后一项中分解出 $A\overline{B}$：

$$BC \cdot 1 + A\overline{B}(\overline{C} + C) + \overline{A}\,\overline{B}\,\overline{C}$$

步骤 3：对上式第一项应用法则 4(舍去 1)，为圆括号中的项应用法则 6($\overline{C} + C = 1$)：

$$BC + A\overline{B} \cdot 1 + \overline{A}\,\overline{B}\,\overline{C}$$

步骤 4：对上式第二项应用法则 4(舍去 1)：

$$BC + A\overline{B} + \overline{A}\,\overline{B}\,\overline{C}$$

步骤 5：从上式第二项和第三项中分解出 $\overline{B}$：

$$BC + \overline{B}(A + \overline{A}\,\overline{C})$$

步骤 6：对上式圆括号中的项应用法则 11($A + \overline{A}\overline{C} = A + \overline{C}$)：

$$BC + \overline{B}(A + \overline{C})$$

步骤 7：对上式使用分配和交换定律得到下面的表达式：

$$BC + A\overline{B} + \overline{B}\,\overline{C}$$

相关问题：化简布尔表达式 $AB\overline{C} + \overline{A}BC + \overline{A}BC + \overline{A}B\overline{C}$。

例 4.11　化简下面的布尔表达式：

$$\overline{AB + AC} + \overline{A}\,\overline{B}C$$

解：

步骤 1：对上式应用狄摩根定理：

$$(\overline{AB})(\overline{AC}) + \overline{A}\,\overline{B}C$$

步骤 2：对上式圆括号中的每一项都应用狄摩根定理：

$$(\overline{A} + \overline{B})(\overline{A} + \overline{C}) + \overline{A}\,\overline{B}C$$

步骤 3：对上式圆括号中的两项应用分配律：

$$\overline{A}\,\overline{A} + \overline{A}\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{B}C$$

步骤 4：对上式第一项应用法则 7($\overline{A}\overline{A} = \overline{A}$)，然后为第三项和最后一项应用法则 10[$\overline{A}\overline{B} + \overline{A}\overline{B}C = \overline{A}\overline{B}(1 + C) = \overline{A}\overline{B}$]：

$$\overline{A} + \overline{A}\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$$

步骤 5:对上式第一项和第二项应用法则 10[$\overline{A} + \overline{A}\overline{C} = \overline{A}(1 + \overline{C}) = \overline{A}$]:

$$\overline{A} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$$

步骤 6:对上式第一项和第二项应用法则 10[$\overline{A} + \overline{A}\overline{B} = \overline{A}(1 + \overline{B}) = \overline{A}$]:

$$\overline{A} + \overline{B}\,\overline{C}$$

相关问题:简化布尔表达式$\overline{AB} + \overline{AC} + \overline{A}B\overline{C}$。

4.6 布尔表达式的标准形式

所有的布尔表达式,不管它们的形式如何,都可以变换为两种标准形式之一:乘积项之和及其和项之乘积。标准化使得计算、化简及实现布尔表达式都变得更为系统和简便。

学完本节以后,应当能够

- 认识表达式乘积项之和
- 确定布尔表达式的域
- 把任何乘积项之和表达式转换为标准形式
- 以二进制数值计算最小项(标准乘积项)之和表达式
- 认识和项之乘积表达式
- 把和项之乘积表达式转换为标准形式
- 以二进制数值计算最大项(标准和项)之积表达式
- 将一种标准形式转换为另一种标准形式

4.6.1 乘积项之和形式

◇ 乘积项之和表达式可以由一个或门及两个或者更多的与门实现。

在 4.1 节中,乘积项定义为由文字(变量或者它们的反码)的乘积(布尔乘法)组成的项。当两个或者更多个乘积项由布尔加法加起来时,得到的表达式就是乘积项之和(SOP)。其中的一些例子是

$$AB + ABC$$
$$ABC + CDE + \overline{B}C\overline{D}$$
$$\overline{A}B + \overline{A}B\overline{C} + AC$$

当然,一个乘积项之和表达式可以包含单变量项,比如 $A + \overline{A}\overline{B}C + BC\overline{D}$。参见上一节的化简的例子,就会发现每一个最终表达式都是一个单个的乘积项或者是乘积项之和的形式。在乘积项之和的表达式中,单个上划杠不能延伸到一个以上的变量;但是,一个项中的多个变量可以分别拥有一个上划杠。例如,可以有 $\overline{A}\,\overline{B}\,\overline{C}$ 项,而不是$\overline{ABC}$。

布尔表达式的变量域 一般布尔表达式的域是该表达式中所包含的变量集合,可以是反码或原码的形式。例如,表达式 $\overline{A}B + A\overline{B}C$ 的域就是变量 A、B、C 的集合,而表达式 $AB\overline{C} + C\overline{D}E + \overline{B}C\overline{D}$ 的域就是变量 A、B、C、D、E 的集合。

乘积项之和表达式的与/或实现　实现乘积项之和表达式仅仅需要将两个或者更多与门的输出进行或运算就可以了。乘积项由与运算产生,两个或者多个乘积项的和(加)由一个或运算完成。所以,乘积项之和表达式可以由与 - 或逻辑来实现,其中与门(个数等于表达式中乘积项的个数)的输出连接到一个或门的输入上,如图 4.18 所展示的 $AB + BCD + AC$ 表达式。或门的输出 X 就等于该乘积项之和表达式。

乘积项之和表达式的与非/与非实现　与非门可以用来实现乘积项之和表达式。如图 4.19所给出的,仅仅使用与非门可以完成与/或的功能。第一级与非门的输出连接到以非 - 或门形式出现的与非门上。第一级与非门输出的反相和非 - 或门输入的反相抵消,最后的结果实际上就是一个与/或电路。

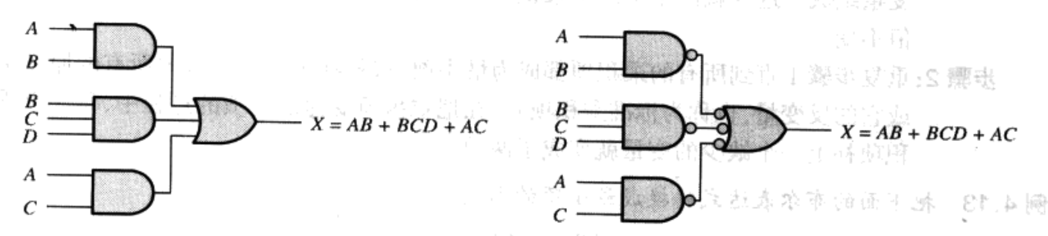


图 4.18　乘积项之和表达式 $AB + BCD + AC$ 的实现　　图 4.19　与非/与非的实现相当于图 4.18 中的与/或

4.6.2　一般表达式向乘积项之和形式的变换

应用布尔代数技术,任何的逻辑表达式都可以变换为乘积项之和的形式。例如,表达式 $A(B + CD)$就可以使用分配律而变换为乘积项之和的形式:

$$A(B + CD) = AB + ACD$$

例 4.12　把下面的每个布尔表达式转换为乘积项之和的形式:

(a) $AB + B(CD + EF)$　　(b) $(A + B)(B + C + D)$　　(c) $\overline{(A + \bar{B}) + C}$

解:

(a) $AB + B(CD + EF) = AB + BCD + BEF$

(b) $(A + B)(B + C + D) = AB + AC + AD + BB + BC + BD$

(c) $\overline{(A + \bar{B}) + C} = (\overline{\overline{A + B}})\bar{C} = (A + B)\bar{C} = A\bar{C} + B\bar{C}$

相关问题: 把 $\bar{A}B\bar{C} + (A + \bar{B})(B + \bar{C} + A\bar{B})$转换为乘积项之和的形式。

4.6.3　最小项(标准乘积项)之和的形式

到目前为止,我们已经看到了乘积项之和表达式,其中有一些乘积项并没有包含表达式域中的所有变量。例如,表达式 $\bar{A}B\bar{C} + A\bar{B}D + \bar{A}B\bar{C}D$的域由变量 A、B、C 和 D 组成。但是,注意在表达式的前两项中并没有出现域的全部变量的集合:也就是说,第一项缺少了 D 或 $\bar{D}$,第二项缺少了 C 或 $\bar{C}$。

最小项(标准乘积项)之和表达式的每一个乘积项都包含该表达式域中的所有变量:$A\bar{B}CD + \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D}$ 就是一个最小项之和表达式。最小项之和表达式在构建真值表 (4.7 节中介

绍)中是非常重要的,并且在卡诺(Karnaugh)图中的化简(在4.8节介绍)也是非常重的。任何非标准的乘积项之和表达式(简单看做乘积项之和)都可以使用布尔代数把它变换为最小项之和的形式。

把乘积项转换为最小项 乘积项之和表达式中没有包含域的所有变量的每一个乘积项,都可以扩展成最小项以包含域中所有的变量或反变量。如下面的步骤所表述的一样,使用表4.1中的布尔代数法则6($A+\overline{A}=1$):变量加上它的反码等于1,可以把非标准乘积项表达式转换为最小项表达式。

步骤1: 对每个非标准乘积项乘上一项,这一项由此乘积项中没有出现的变量加上它的反变量组成。这样就产生了两个乘积项。如所知道的那样,任何乘积项乘以1,它的值不变。

步骤2: 重复步骤1直到所有的乘积项都成为最小项(乘积项中包含了域中所有的原变量或它的反变量,也称为标准乘积项)。在把乘积项变为最小项的过程中,每一个乘积项补上一个缺少的变量就变成了两项。

例4.13 把下面的布尔表达式转换成最小项的形式:

$$A\overline{B}C+\overline{A}\,\overline{B}+AB\overline{C}D$$

解: 乘积项表达式的域是A、B、C、D。每次取一项,第一项$A\overline{B}C$缺少变量D或$\overline{D}$,所以用$D+\overline{D}$乘第一项,如下所示:

$$A\overline{B}C=A\overline{B}C(D+\overline{D})=A\overline{B}CD+A\overline{B}C\overline{D}$$

在此情况下,结果就是两个最小项。

第二项$\overline{A}\,\overline{B}$缺少变量$C$或$\overline{C}$及$D$或$\overline{D}$,所以先用$C+\overline{C}$乘第二项,如下所示:

$$\overline{A}\,\overline{B}=\overline{A}\,\overline{B}(C+\overline{C})=\overline{A}\,\overline{B}C+\overline{A}\,\overline{B}\,\overline{C}$$

上式得到的两个乘积项缺少D或$\overline{D}$,所以用$D+\overline{D}$乘这两项,如下所示:

$$\begin{aligned}\overline{A}\,\overline{B}&=\overline{A}\,\overline{B}C+\overline{A}\,\overline{B}\,\overline{C}=\overline{A}\,\overline{B}C(D+\overline{D})+\overline{A}\,\overline{B}\,\overline{C}(D+\overline{D})\\&=\overline{A}\,\overline{B}CD+\overline{A}\,\overline{B}C\overline{D}+\overline{A}\,\overline{B}\,\overline{C}D+\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\end{aligned}$$

在此例子中,结果得到四个最小项。

第三项$AB\overline{C}D$已经是最小项了。原始表达式的完整的最小项之和的表达式如下:

$$A\overline{B}C+\overline{A}\,\overline{B}+AB\overline{C}D=A\overline{B}CD+A\overline{B}C\overline{D}+\overline{A}\,\overline{B}CD+\overline{A}\,\overline{B}C\overline{D}+\overline{A}\,\overline{B}\,\overline{C}D+\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}+AB\overline{C}D$$

相关问题: 把表达式$W\overline{X}Y+\overline{X}Y\overline{Z}+WX\overline{Y}$转换为最小项之和的形式。

最小项的二进制表达 一个最小项等于1,这时仅对应一种二进制变量值的组合。例如,乘积项$A\overline{B}C\overline{D}$等于1,这时$A=1$、$B=0$、$C=1$、$D=0$,如下所示,对于其他二进制变量值的组合都等于0。

$$A\overline{B}C\overline{D}=1\cdot\overline{0}\cdot1\cdot\overline{0}=1\cdot1\cdot1\cdot1=1$$

在这里,乘积项的二进制值是1010(十进制数10)。

记住,乘积项是由与门实现的,只有在与门的输入都是1的情况下输出才为1。反相器用来产生所需要的变量的反码。

一个乘积项之和的表达式等于1,只有在表达式中有一个或多个乘积项等于1时才成立。

例 4.14　已知下面最小项之和的表达式等于 1,确定每个最小项的二进制值:

$$ABCD + A\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

解: $ABCD$ 项等于 1,这时 $A=1$、$B=1$、$C=1$ 和 $D=1$:

$$ABCD = 1\cdot1\cdot1\cdot1 = 1$$

$A\overline{B}\,\overline{C}D$ 项等于 1,这时 $A=1$、$B=0$、$C=0$ 和 $D=1$:

$$A\overline{B}\,\overline{C}D = 1\cdot\overline{0}\cdot\overline{0}\cdot1 = 1\cdot1\cdot1\cdot1 = 1$$

$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ 项等于 1,这时 $A=0$、$B=0$、$C=0$ 和 $D=0$:

$$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} = \overline{0}\cdot\overline{0}\cdot\overline{0}\cdot\overline{0} = 1\cdot1\cdot1\cdot1 = 1$$

乘积项之和的表达式等于 1,只有在表达式中有一个或多个乘积项等于 1 时才成立。

相关问题: 已知下面最小项之和的表达式等于 1,确定每个最小项的二进制值:

$$\overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + \overline{X}Y\overline{Z} + XYZ$$

这是否是最小项之和表达式?

4.6.4　和项之乘积形式

在 4.1 节中,和项定义为文字(变量或者它们的反码)的和(布尔加)项。当两个或者更多的和项相乘时,结果表达式就是和项之乘积(POS)。例如:

$$(\overline{A} + B)(A + \overline{B} + C)$$
$$(\overline{A} + \overline{B} + \overline{C})(C + \overline{D} + E)(\overline{B} + C + D)$$
$$(A + B)(A + \overline{B} + C)(\overline{A} + C)$$

一个和项之乘积的表达式可以包含单变量项,如 $\overline{A}(A + \overline{B} + C)(\overline{B} + \overline{C} + D)$。在和项之乘积的表达式中,一个单一的上划杠不能扩展到多于一个的变量。例如,一个和项之乘积表达式中可以有 $\overline{A} + \overline{B} + \overline{C}$ 项,但是不可以有 $\overline{A + B + C}$ 项。

和项之乘积的实现　实现和项之乘积表达式,只需简单地把两个或多个或门的输出进行与运算。和项由或运算产生,两个或多个和项的乘积由一个与运算产生。因此,一个和项之乘积表达式可以由若干个或门(和项表达式中和项的数目相同)的输出连接到一个与门的输入这样的逻辑电路来实现。图 4.20 给出了实现表达式 $(A + B)(B + C + D)(A + C)$ 的电路。与门的输出 X 等于和项之乘积表达式。

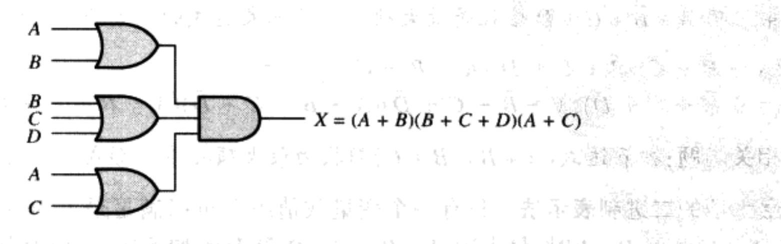


图 4.20　和项之乘积表达式 $(A + B)(B + C + D)(A + C)$ 的实现

4.6.5 最大项(标准和项)之积形式

到目前为止,我们已经看到了 POS 表达式,其中有些和项并没有包含表达式域中的所有变量,例如表达式:

$$(A+\overline{B}+C)(A+B+\overline{D})(A+\overline{B}+\overline{C}+D)$$

它的域由变量 A、B、C、D 组成。注意,在表达式的前两项没有出现域中的全部变量的集合,也就是说,第一项缺少了 D 或者 $\overline{D}$,第二项缺少了 C 或者 $\overline{C}$。

最大项之积的表达式中每一个和项都包含域中的所有变量。例如,表达式:

$$(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+\overline{B}+C+D)(A+B+\overline{C}+D)$$

就是一个最大项之乘积表达式。使用布尔表代数,任何非最大项之乘积表达式(简单地看做和项之乘积)可以转换成最大项之乘积表达式。

和项转换为最大项(标准和项) 在和项之乘积表达式中,没有包含域中所有变量的每一个和项,都可以扩展到最大项之积的形式,以包括域中的所有变量及它们的反码。正如在下面的步骤所陈述的,可以使用表 4.1 中的布尔代数法则 8($A\cdot\overline{A}=0$):一个变量乘以它的反码等于 0,把和项之乘积表达式变换为最大项之积的形式。

步骤 1: 为每一个非标准和项加上一个由缺少的变量和它的补码的乘积构成的项,结果就产生两个和项。如同所知道的那样,在任意数上加 0 都不会改变这个数的值。

步骤 2: 应用表 4.1 中的法则 12:[$A+BC=(A+B)(A+C)$]。

步骤 3: 重复步骤 1 直到所有的和项都包含域中的所有变量,这些变量可以是原码或是反码。

例 4.15 把下面的布尔表达式转换为最大项之乘积的形式:

$$(A+\overline{B}+C)(\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$$

解: 表达式的域是 A、B、C、D。每次取一个项。第一项 $A+\overline{B}+C$,缺少了 D 或者 $\overline{D}$,所以添加 $D\overline{D}$ 并使用法则 12,如下所示:

$$A+\overline{B}+C=A+\overline{B}+C+D\overline{D}=(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})$$

第二项缺少了变量 A 或者 $\overline{A}$,所以加上 $A\overline{A}$ 并且应用法则 12,如下所示:

$$\overline{B}+C+\overline{D}=\overline{B}+C+\overline{D}+A\overline{A}=(A+\overline{B}+C+\overline{D})(\overline{A}+\overline{B}+C+\overline{D})$$

第三项 $A+\overline{B}+\overline{C}+D$ 已经是最大项了。原始表达式的最大项之乘积表达式如下所示:

$$(A+\overline{B}+C)(\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)=$$
$$(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})(A+\overline{B}+C+\overline{D})(\overline{A}+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$$

相关问题: 把表达式$(A+\overline{B})(B+C)$转换为最大项之乘积形式。

最大项的二进制表示法 只有一个变量数值组合可以满足最大项等于 0。例如,当 $A=0$、$B=1$、$C=0$ 及 $D=1$ 时,最大项 $A+\overline{B}+C+\overline{D}$ 等于 0,如下所示,所有其他变量的值的组合都是 1。

$$A+\overline{B}+C+\overline{D}=0+\overline{1}+0+\overline{1}=0+0+0+0=0$$

在这里，和项的二进制值是 0101（十进制 5）。记住，和项由一个或门实现，只有或门的每一个输入都是 0 时，输出才是 0。反相器用来产生所需变量的反码。

只有当表达式中的一个或者多个和项等于 0 时，和项之乘积表达式才等于 **0**。

例 4.16　确定变量的二进制数值，使得下面的最大项之和的表达式等于 0：

$$(A+B+C+D)(A+\overline{B}+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})$$

解：当 $A=0$、$B=0$、$C=0$ 和 $D=0$ 时，项 $A+B+C+D$ 等于 0：

$$A+B+C+D=0+0+0+0=0$$

当 $A=0$、$B=1$、$C=1$ 和 $D=0$ 时，项 $A+\overline{B}+\overline{C}+D$ 等于 0：

$$A+\overline{B}+\overline{C}+D=0+\overline{1}+\overline{1}+0=0+0+0+0=0$$

当 $A=1$、$B=1$、$C=1$ 和 $D=1$ 时，项 $\overline{A}+\overline{B}+\overline{C}+\overline{D}$ 等于 0：

$$\overline{A}+\overline{B}+\overline{C}+\overline{D}=\overline{1}+\overline{1}+\overline{1}+\overline{1}=0+0+0+0=0$$

当三个最大项中的任何一项等于 0 时，该最大项表达式就等于 0。

相关问题：确定变量的二进制数值，使得下面的和项之乘积表达式等于 0：

$$(X+\overline{Y}+Z)(\overline{X}+Y+Z)(X+Y+\overline{Z})(\overline{X}+\overline{Y}+\overline{Z})(X+\overline{Y}+\overline{Z})$$

这是个最大项表达式吗？

4.6.6　把最小项之和（SOP）转换为最大项之积（POS）

给定的最小项之和表达式中乘积项的二进制数值，并不会出现在对应的最大项之积表达式中。同样，最小项之和表达式中没有出现的二进制数值却出现在对应的最大项之积表达式中。因此，为了把将最小项之和转换为最大项之积，可以采取下面的步骤：

步骤 1：计算乘积项之和表达式中的每一个乘积项，也就是说，确定表示乘积项的二进制数；
步骤 2：确定步骤 1 中的计算没有包含的所有二进制数；
步骤 3：为从步骤 2 得到的每一个二进制数写出相应的和项，并以和项之乘积形式表达。

使用相似的过程，就可以把和项之乘积转换为乘积项之和。

例 4.17　把下面的最小项之和表达式转换为等价的最大项之积表达式：

$$\overline{A}\,\overline{B}\,\overline{C}+\overline{A}B\overline{C}+\overline{A}BC+A\overline{B}C+ABC$$

解：计算如下所示：

$$000+010+011+101+111$$

因为表达式的域有 3 个变量，所有总共有 $8(2^3)$ 个可能的组合。最小项之和表达式包含这些组合中的 5 个，因此，最大项之积表达式必须包含其余的三个，它们是 001、100 和 110。记住，这些二进制数使得和项等于 0。等价的乘积项之和表达式为

$$(A+B+\overline{C})(\overline{A}+B+C)(\overline{A}+\overline{B}+C)$$

相关问题：通过在每个表达式中代入二进制数值，验证这个例中的最小项之和表达式和最大项之积表达式是等价的。

4.7 布尔表达式和真值表

为表达式中的每一项都使用二进制数值，那么所有的标准布尔表达式都可以很容易地变换为真值表的形式。真值表是以简明的形式表示电路逻辑运算的常用方法。当然，最小项之和表达式或最大项之积表达式可以从真值表中确定。可以在数据表及与数字电路运算有关的其他文字中发现真值表。

学完本节以后，应当能够

- 把最小项之和表达式转换为真值表的形式
- 把最大项之积表达式转换为真值表的形式
- 从真值表中导出标准表达式
- 正确解释真值表数据

4.7.1 把乘积项之和表达式转换为真值表的形式

记得在4.6节中，只有在至少有一个乘积项等于1时，乘积项之和表达式才等于1。真值表只是简单地列出了输入变量数值的可能组合及相应的输出数值(1或者0)。对于域有两个变量的表达式，那么这两个变量有$4(2^2=4)$种不同的组合。对于域有3个变量的表达式，这些变量有$8(2^3=8)$个不同的组合。对于域有4个变量的表达式，这些变量有$16(2^4=16)$个不同的组合，以此类推。

构建真值表的第一步，是列出表达式中变量的所有可能的二进制数值组合。接下来，如果乘积项之和表达式不是最小项之和表达式，那么把它转换为最小项之和表达式。最后，对每个使得最小项之和表达式为1的二进制数，在其输出列(X)的相对应的地方放置一个1，为所有剩余的二进制数值对应的输出位置上置0。这个过程如例4.18所示。

例4.18 为最小项之和表达式 $\overline{A}\overline{B}C+A\overline{B}\overline{C}+ABC$ 写出一个真值表。

解：在这个域中有3个变量，所以有8种可能的变量的二进制值的组合，它们列在了表4.6左边的3列上。使得表达式中乘积项等于1的二进制数是 $\overline{A}\overline{B}C$:001；$A\overline{B}\overline{C}$:100；$ABC$:111。对每一个这样的二进制数，在输出列的相应位置放置一个1，如表中所示。对于剩余的二进制数的组合，则在输出列上放置0。

相关问题：为最小项之和表达式 $\overline{A}B\overline{C}+A\overline{B}C$ 创建真值表。

4.7.2 把和项之乘积表达式转换为真值表的形式

回顾一下，只有在至少有一个和项等于0时，和项之乘积项表达式才等于0。为了从和项之乘积项表达式中构建真值表，可以像最小项之和表达式那样，列出所有可能的变量的二进制数组合，如果这个和项之乘积表达式不是最大项之积表达式，那么将该表达式转换为最大项之

积表达式的形式。最后,使得最大项之积表达式等于 0 的二进制数相对应的输出列(X)上放置一个 0,为所有剩余的二进制数值对应的输出列上放置 1。这个过程如例 4.19 所示。

表 4.6

输入			输出	
A	B	C	X	乘积项
0	0	0	0	
0	0	1	1	$\overline{A}\,\overline{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

例 4.19　为下面的最大项之和表达式确定其真值表:

$$(A+B+C)(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+C)$$

解: 这个域具有 3 个变量,8 个可能的二进制数值列在了表 4.7 左边的三列。使得表达式中和项的二进制数是 $A+B+C$:000; $A+\overline{B}+C$:010; $A+\overline{B}+\overline{C}$:011; $\overline{A}+B+\overline{C}$:101; $\overline{A}+\overline{B}+C$:110。为这些二进制数值的每一个,在相应的输出列中都置入 0,如表中所示。对于其余的二进制数值的组合,则在相应的输出列中置入 1。

表 4.7

输入			输出	
A	B	C	X	和项
0	0	0	0	$(A+B+C)$
0	0	1	1	
0	1	0	0	$(A+\overline{B}+C)$
0	1	1	0	$(A+\overline{B}+\overline{C})$
1	0	0	1	
1	0	1	0	$(\overline{A}+B+\overline{C})$
1	1	0	0	$(\overline{A}+\overline{B}+C)$
1	1	1	1	

注意这个例中的真值表和例 4.18 中的真值表是一样的。这就说明前面例中的乘积项之和表达式和此例中的和项之乘积表达式是等价的。

相关问题: 为下面的最大项之积表达式开发真值表:

$$(A+\overline{B}+C)(A+B+\overline{C})(\overline{A}+\overline{B}+\overline{C})$$

4.7.3 从真值表确定标准表达式

为了确定真值表表示的最小项之和表达式,列出使得输出为 1 的输入变量的二进制数。把乘积项中相应的变量替换成 1,反变量替换成 0,这样就把每一个二进制数替换成相应的乘积项。例如,二进制数 1010 可以转换成如下所示的乘积项:

$$1010 \longrightarrow A\overline{B}C\overline{D}$$

代入以后,就会发现这个乘积项为 1:

$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

为了确定真值表所表示的最大项之积表达式,列出使得输出为 0 的二进制数。把和项中相应的变量替换成 0,反变量替换成 1,这样就把每一个二进制数替换成相应的和项。例如,二进制数 1001 可以转换成如下所示的和项:

$$1001 \longrightarrow \overline{A} + B + C + \overline{D}$$

代入以后,就会发现这个和项为 0:

$$\overline{A} + B + C + \overline{D} = \overline{1} + 0 + 0 + \overline{1} = 0 + 0 + 0 + 0 = 0$$

例 4.20 从表 4.8 中的真值表中,确定最小项之和表达式及等价的最大项之积表达式。

表 4.8

输入			输出
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

解: 在输出列中有 4 个 1,并且相应的二进制数是 011、100、110 及 111。把这些二进制数转换为乘积项,如下所示:

$$011 \longrightarrow \overline{A}BC$$
$$100 \longrightarrow A\overline{B}\,\overline{C}$$
$$110 \longrightarrow AB\overline{C}$$
$$111 \longrightarrow ABC$$

最后,输出 X 的最小项之和表达式是

$$X = \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$$

对于最大项之积表达式,二进制值 000、001、010 和 101 的输出为 0。把这些二进制数转换为和项,如下所示:

$$000 \longrightarrow A + B + C$$
$$001 \longrightarrow A + B + \overline{C}$$
$$010 \longrightarrow A + \overline{B} + C$$
$$101 \longrightarrow \overline{A} + B + \overline{C}$$

最后，输出 X 的最大项之积表达式是

$$X = (A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(\overline{A} + B + \overline{C})$$

相关问题：通过代入二进制数，展示这个例中所导出的最小项之和与最大项之积表达式是等价的。也就是说，对于任何一个二进制数值，它们应当都是 1 或者都是 0，结果取决于该二进制数值。

4.8　卡诺图

卡诺(Karnaugh)图提供了简化布尔表达式的一种系统方法，如果使用正确，将会得到尽可能简化的乘积项之和或和项之乘积表达式，也就是所谓的最小表达式。正所我们看到的那样，代数简化效果的优劣取决于对布尔代数的所有定律、法则及定理的熟悉程度，同时还取决于应用能力。从另一方面来说，卡诺图提供了用以简化的“烹饪书”。

学完本节以后，应当能够

- 为 3 个或者 4 个变量构建卡诺图
- 确定卡诺图中每一小方块的二进制数值
- 确定由卡诺图中每一小方格所表示的最小项
- 解释小方块邻接并识别相邻项

◇ 卡诺图的目的是用以简化布尔表达式。

卡诺图和真值表的相似之处，是因为它呈现了所有可能的输入变量的数值，以及这些值所对应的输出。和真值表组织成列和行的样式不同，卡诺图是小方格的阵列，其中每一个小方格表示一个输入变量的二进制数值。这种布置小方格的方式，使得对给定表达式的简化仅仅是把小方格恰当地进行组合的事情。卡诺图可以用于具有 2 个、3 个、4 个和 5 个变量的表达式，但是这里仅仅讨论 3 变量和 4 变量的情况，用以阐释该原理。4.11 节使用 32 个小方格的卡诺图去处理 5 个变量。另一种超出本书讨论范围的方法，称为奎恩 - 麦克拉斯基(Quine-McClusky)化简方法，可以用于有较多变量的情况。

卡诺图中小方格的数目等于可能输入变量组合的总数目，也就是真值表中行的数目。对于 3 个变量，小方格的数目是 $2^3 = 8$。对于 4 个变量，小方格的数目为 $2^4 = 16$。

4.8.1　3 变量卡诺图

3 变量卡诺图是 8 个小方格阵列，如图 4.21(a)所示。在这种情况下，A、B、C 用以表示变量，当然也可以使用其他的字母。A 和 B 的二进制数值沿着左边界(注意次序)而 C 的数横跨顶部。一个给定小方格的数值就是同一行中左边的 A 和 B 的数值及同一列中顶部 C 的值。例如，左上角小方格的二进制数值为 000，右下角小方格的二进制数值为 101。图 4.21(b)给出了卡诺图中每一个小方格所表示的最小项。

4.8.2 4变量卡诺图

4变量卡诺图是16个小方格的阵列，如图4.22(a)所示。A和B的二进制数值沿着左边界，C和D的数值横跨顶部。一个给定小方格的值就是同一行左边的A和B数值及同一列顶部的C和D数值。例如，右上角小方格的二进制数值为0010，右下角单元的二进制数为1010。图4.22(b)展示了4变量卡诺图每一个单元所表示的最小项。

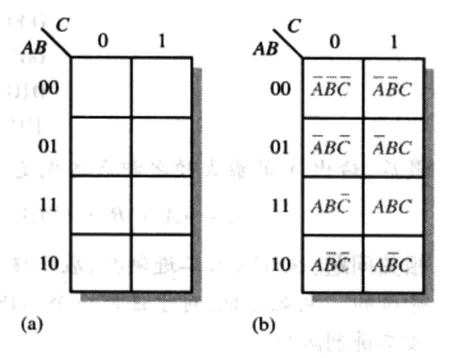


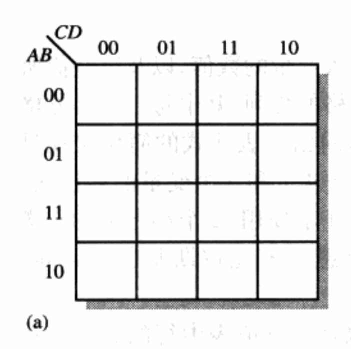
图4.21 展示乘积项的3变量卡诺图

4.8.3 小方格相邻

◇ 仅有一个变量不同的小方格相邻。
◇ 有一个以上变量不同的小方格不相邻。

卡诺图中的小方格按一定方式布置，使得相邻的小方格之间只有其中一个变量发生变化。相邻(adjacency)定义为其中一个变量发生变化。在3变量图中，010小方格和000、011及110小方格相邻。010小方格和001、111、100或101小方格不相邻。

从物理上来看，每一个小方格都和与它有4条边紧挨着的小方格相邻。一个小方格和与它有对角接触的小方格不相邻。同样，在顶行的小方格和底部一行相应的小方格相邻，左上角的小方个格和右下角的小方格相邻。这称为“环绕”相邻，因为可以将这个图从顶部到底部环绕而形成一个圆柱体，或者从左到右环绕而形成一个圆柱体。图4.23给出了4变量的卡诺图的小方格相邻，然而相邻法则同样适用于任意数目的小方格。



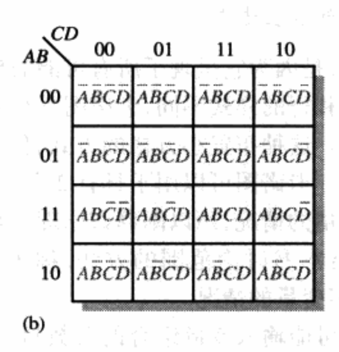


图4.22 4变量卡诺图

图4.23 卡诺图上的相邻小方格仅有一个变量不同。在相邻的小方格之间由箭头标出

4.9 卡诺图乘积项之和的最小化

如上一节所表述的那样，卡诺图用以将布尔表达式化简到它们的最小形式。一个最小化的乘积项之和表达式包含尽可能少的项，并且每一项中包含尽可能少的变量。一般来说，一个最小的乘积项之和表达式可以用比标准表达式更少的逻辑门来实现。

学完本节之后,应该能够

- 在卡诺图上映射最小项之和表达式
- 把卡诺图上的 1 合并成最大的组
- 从卡诺图上的每一个最大组确定其最小的乘积项
- 把最小的乘积项相加以形成最小乘积项之和表达式
- 把真值表转换为卡诺图,用以对所给出的表达式进行化简
- 在卡诺图上使用“无关项”条件

4.9.1 最小项之和表达式的卡诺图映射

对于最小项之和表达式来说,为表达式中的每一个乘积项都在卡诺图上放置一个 1。每一个 1 都被放置在与乘积项数值相对应的小方格内。例如,对于乘积项 $A\overline{B}C$,一个 1 就被放置在 3 变量卡诺图的 101 小方格内。

当乘积项之和表达式被完全映射好之后,那么卡诺图上就会有一些个 1,其数目等于最小项之和表达式中乘积项的个数。不具有 1 的小方格就是表达式为 0 时所对应的小方格。通常,使用乘积项之和表达式时,0 并不在图上绘出。下面的步骤及图 4.24 中的解释给出了映射卡诺图的过程。

步骤 1: 确定最小项之和表达式中每一个乘积项的二进制数值,这通常可以心算;

步骤 2: 算完每一个乘积项之后,在卡诺图中和乘积项具有相同数值的小方格处放置 1。

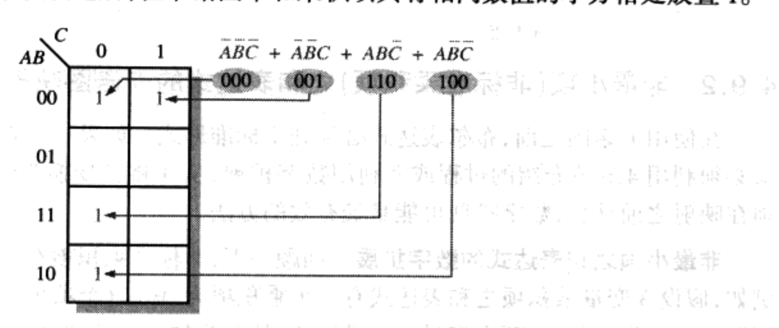


图 4.24 最小项之和表达式映射的例子

下面的例子将进一步阐述映射过程。

例 4.21 把下面的最小项之和表达式映射到卡诺图上:

$$\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$

解: 表达式的计算如下所示。在图 4.25 中的 3 变量卡诺图中,为表达式中每一个最小项都放置 1。

$$\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$

$$001 \quad 010 \quad 110 \quad 111$$

相关问题: 把最小项之和表达式 $\overline{A}BC + A\overline{B}C + A\overline{B}\,\overline{C}$ 映射到卡诺图上。

例 4.22 把下面的最小项之和表达式映射到卡诺图上。

$$\overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}C\overline{D}$$

解: 表达式的计算如下所示。在图 4.26 中的 4 变量卡诺图上,为表达式中每一个最小项都放置一个 1。

$$\begin{matrix} \overline{A}\,\overline{B}CD & + \overline{A}B\overline{C}\,\overline{D} & + AB\overline{C}D & + ABCD & + AB\overline{C}\,\overline{D} & + \overline{A}\,\overline{B}\,\overline{C}D & + A\overline{B}C\overline{D} \\ 0011 & 0100 & 1101 & 1111 & 1100 & 0001 & 1010 \end{matrix}$$

相关问题: 把下面的最小项之和表达式映射到卡诺图上。

$$\overline{A}BC\overline{D} + ABC\overline{D} + AB\overline{C}D + ABCD$$

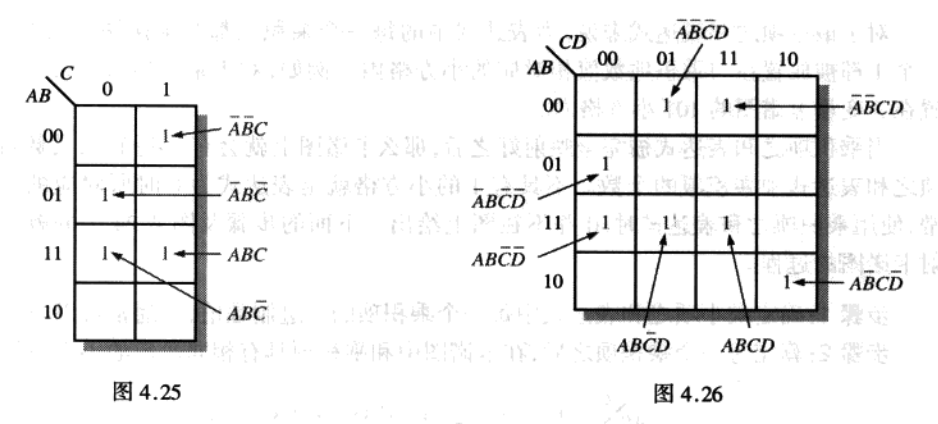


图 4.25　　图 4.26

4.9.2 非最小项(非标准乘积项)之和表达式的卡诺图映射

在使用卡诺图之前,布尔表达式必须处于标准形式。如果一个表达式为非标准形式,那么就必须利用 4.6 节介绍的过程或者利用数字扩展,从而将其变换为标准形式。由于表达式必须在映射之前计算,数字扩展可能是最有效的方法。

非最小项之和表达式的数字扩展　回顾一下,非标准乘积项有一个或者多个缺少的变量。例如,假设 3 变量乘积项之和表达式有一个乘积项为 $A\overline{B}$,这个项可以数字扩展为标准形式,如下所述。首先,写出这两个变量的二进制值,然后为缺少的变量 $\overline{C}$ 添加 0,得到 100。下一步,写出这两个变量的二进制值,然后为缺少的变量 C 添加 1,得到 101。最后得到的这两个二进制数就是最小项(标准乘积项)$A\overline{B}\,\overline{C}$ 和 $A\overline{B}C$ 的值。

作为另一个例子,假设 3 变量表达式的某个乘积项为 B(记住一个单变量可以在乘积项之和表达式中作为一个乘积项)。这个项可以数字扩展到标准项,如下所示。写出该变量的二进制值,然后添加缺少的变量 A 和 C 的所有可能值:

B
010
011
110
111

最后得到的 4 个二进制数就是最小项(标准乘积项)$\overline{A}B\overline{C}$、$\overline{A}BC$、$AB\overline{C}$、$ABC$ 的值。

例 4.23 把下面的乘积项之和表达式映射到卡诺图上：$\overline{A}+A\overline{B}+AB\overline{C}$。

解：这个表达式明显不是标准形式，因为并不是每一个都具有三个变量。第一项缺少了两个变量，第二项缺少了一个变量，第三项是标准的。首先将这些项进行数字扩展，如下所示：

$\overline{A}$	$+A\overline{B}$	$+AB\overline{C}$
000	100	110
001	101	
010		
011		

通过在图 4.27 的 3 变量卡诺图中相应的小方格处放置 1，对每一个二进制数进行映射。

相关问题：把乘积项之和表达式 $BC+\overline{A}\overline{C}$ 映射到卡诺图上。

例 4.24 把下面的乘积项之和表达式映射到卡诺图上。

$$\overline{B}\,\overline{C}+A\overline{B}+AB\overline{C}+A\overline{B}C\overline{D}+\overline{A}\,\overline{B}\,\overline{C}D+A\overline{B}CD$$

解：这个表达式明显不是标准形式，因为并不是每一个都具有 4 个变量。第一项和第二项都缺少了两个变量，第三项缺少了一个变量，其余的项都是标准的。首先对这些项进行数字扩展，这包括了缺少变量的所有组合，如下所示：

$\overline{B}\,\overline{C}$	$A\overline{B}$	$+\ AB\overline{C}$	$+\ A\overline{B}C\overline{D}$	$+\ \overline{A}\,\overline{B}\,\overline{C}D$	$+\ A\overline{B}CD$
0000	1000	1100	1010	0001	1011
0001	1001	1101			
1000	1010				
1001	1011				

通过在图 4.28 中 4 变量卡诺图中相应的小方格中放置 1，对每一个二进制数进行映射。注意在扩展的表达式中有些值是重复的。

相关问题：把表达式 $A+\overline{C}D+AC\overline{D}+\overline{A}BC\overline{D}$ 映射到卡诺图上

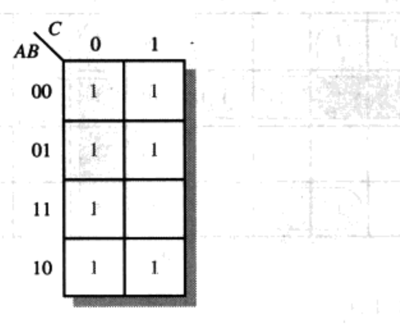


图 4.27

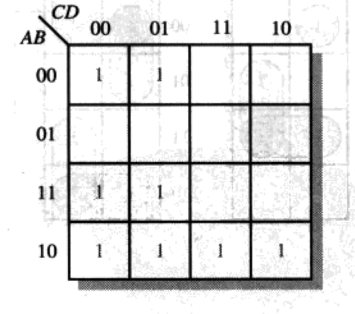


图 4.28

4.9.3 乘积项之和表达式的卡诺图化简

使得表达式包含尽可能少的项，每个项包含尽可能少的变量，这个过程称为最小化。在乘积项之和表达式被映射之后，通过对 1 分组和从图上确定每组的最小乘积项，加起来就得到最小的乘积项之和表达式。

对 1 分组 根据下面的规则,将那些包含 1 的相邻小方格圈在一起,从而在卡诺图上对 1 进行分组。这样做的目的是使得每组(圈)尽可能地大,而组尽可能地少。

1. 一个小组只能包有 1、2、4、8 或者 16 个小方格,全都是 2 的幂。对于 3 变量的卡诺图,最大的组是 $2^3=8$ 个小方格。
2. 小组中的每一个小方格必须与相同小组内的一个或者多个小方格相邻,但是该小组中的所有小方格并不一定相互邻接。
3. 根据规则 1,总是使得所包含的 1 的数目最尽可能地多。
4. 图上的每一个 1 必须包含在至少一个组内。已经在某个组中的 1,也可以包含在其他组中,只要两个重叠的组都含有不属于其他组的 1。

例 4.25 对图 4.29 中的每一个卡诺图上的 1 进行分组。

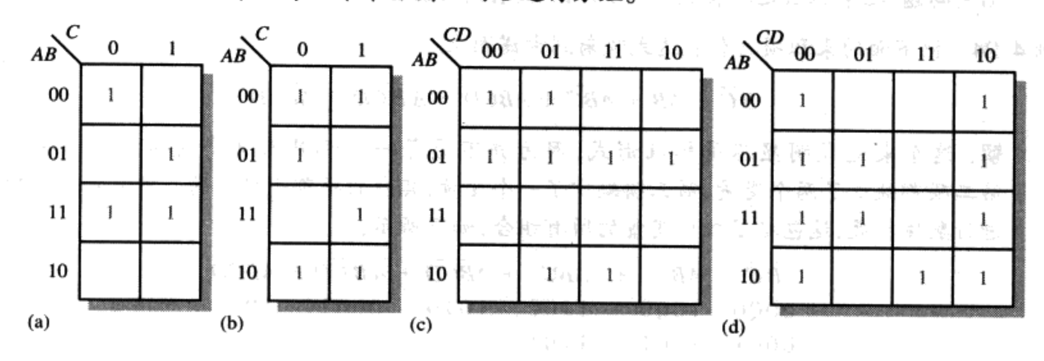


图 4.29

解: 分组过程如图 4.30 所示。在某些情况下,可能具有不止一种的方法对 1 进行分组以形成最大的组。

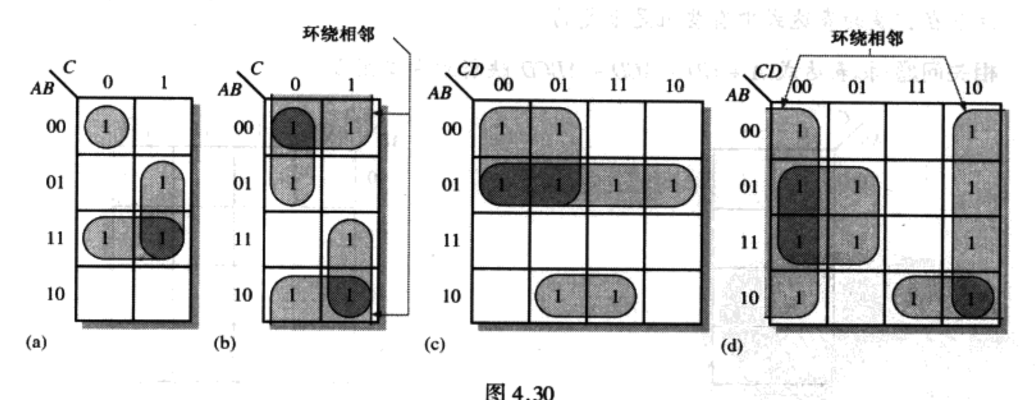


图 4.30

相关问题: 对于图 4.30 中所示的 1,确定是否还有其他的方法进行分组,以得到最大分组情况下的最少的组。

从卡诺图上确定最小乘积项之和表达式 当表达式中所有表示最小项的 1 在卡诺图上得到恰当的映射和分组后,确定最小化的乘积项之和表达式的过程开始。应用下面的规则得到最小化的乘积项和最小化的和项的表达式。

1. 对有 1 的小方格分组。包含 1 的每一组小方格产生一个乘积项，组成这个乘积项的所有变量在这个组中仅出现一种形式(要么是原变量，否则就是反变量)。在这个组中变量发生了从原码到反码的变化，那么删除这些变量。这些变量称为矛盾变量(contradictory variable)。
2. 为每个组确定最小乘积项。
   a. 3 变量的卡诺图
      (1)一个小方格组产生 3 变量的乘积项
      (2)两个小方格组产生 2 变量的乘积项
      (3)四个小方格组产生 1 变量的项
      (4)八个小方格组产生表达式 1
   b. 4 变量的卡诺图
      (1)一个小方格组产生 4 变量的乘积项
      (2)两个小方格组产生 3 变量的乘积项
      (3)四个小方格组产生 2 变量的乘积项
      (4)八个小方格组产生 1 变量的项
      (5)十六个小方格组产生表达式 1
3. 当从卡诺图推导出所有的最小化乘积项后，把它们加起来形成最小乘积项之和表达式。

例 4.26　为图 4.31 中的卡诺图确定乘积项，并写出最小乘积项之和表达式的结果。

解：把小组中原码和反码形式都有的变量删除。在图 4.31 中，八个小方格组的乘积项是 B，因为在这组中的小方中包含了 A 和 $\bar{A}$、C 和 $\bar{C}$ 及 D 和 $\bar{D}$，这些已被删除。四个小方格组包含 B 和 $\bar{B}$、D 和 $\bar{D}$，保留变量 $\bar{A}$ 和 C，形成乘积项 $\bar{A}C$。两个小方格组包含 B 和 $\bar{B}$，保留变量 A、$\bar{C}$ 和 D，形成乘积项 $A\bar{C}D$。注意怎样利用重叠以使得组的规模最大。最后最小乘积项之和表达式就是这些乘积项的和：

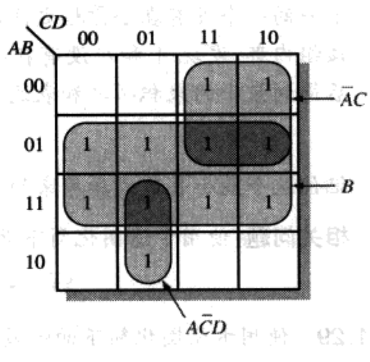


图 4.31

$$B + \bar{A}C + A\bar{C}D$$

相关问题：对于图 4.31 中的卡诺图，在右下角小方格(1010)处添加一个 1，然后确定乘积项之和表达式。

例 4.27　为图 4.32 中的每一个卡诺图确定乘积项，并写出生成的最小化乘积项之和表达式。

解：每一个组得到的最小化乘积项如图 4.32 所示。每个卡诺图的最小化乘积项之和表达式如下：

(a) $AB + BC + \bar{A}\bar{B}\bar{C}$　　(b) $\bar{B} + \bar{A}\bar{C} + AC$

(c) $\bar{A}B + \bar{A}\bar{C} + A\bar{B}D$　　(d) $\bar{D} + A\bar{B}C + B\bar{C}$

相关问题：对图 4.32(d)的卡诺图，在小方格 0111 中放置 1，然后求出乘积项表达式。

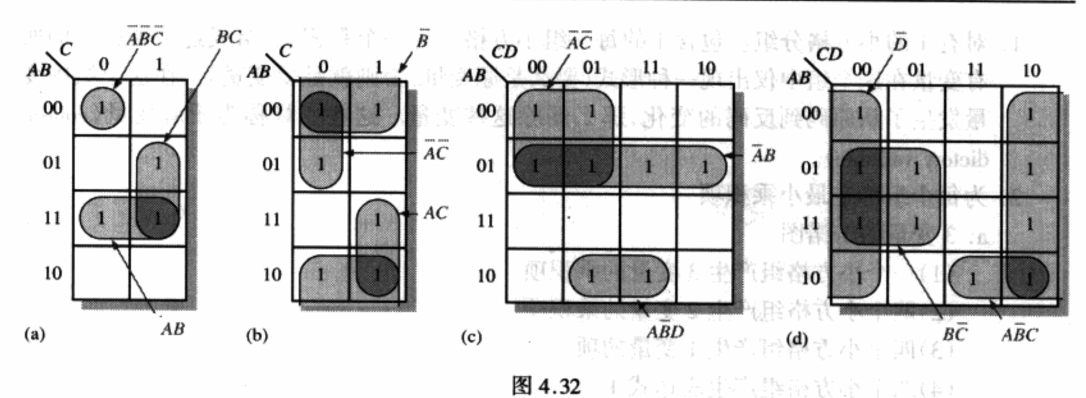


图 4.32

例 4.28 使用卡诺图最小化下面的最小项之和表达式:

$$A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C}$$

解:表达式的二进制值是

$$101 + 011 + 011 + 000 + 100$$

最小项之和表达式的映射和分组后的小方格如图 4.33 所示。

注意那个“环绕”4 个小方格组,包括了顶行及底行中的所有 1。剩下的 1 被吸纳在一个两个小方格的重叠组中。4 个 1 的组产生一个单变量项 $\overline{B}$。通过观察组的内部,发现 $\overline{B}$ 是唯一的一个没有在小方格之间改变的变量。两个 1 的组产生两个变量项 $\overline{A}C$。通过观察该组内部,发现 $\overline{A}$ 和 C 没有在小方格之间改变。每一个小组的乘积项被展示了出来,最后得到最小的乘积项之和表达式为

$$\overline{B} + \overline{A}C$$

记住这个最小表达式和原来的标准表达式是等价的。

相关问题:使用卡诺图化简下面的最小项表达式:

$$X\overline{Y}Z + XY\overline{Z} + \overline{X}YZ + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XYZ$$

例 4.29 使用卡诺图化简下面的最小项表达式:

$$\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}CD + A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$$

解:第一项 $\overline{B}\,\overline{C}\,\overline{D}$ 必须被扩展为 $A\overline{B}\,\overline{C}\,\overline{D}$ 和 $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ 以得到最小项,然后将其映射;分组后的小方格如图 4.34 所示。

注意那两个呈现“环绕”相邻的组。因为作业最外两列中的小方格是相邻的,所以可以形成 8 个 1 的组。由于顶部和底部的小方格是相邻的,所以形成 4 个 1 的组,用以获取剩余的两个 1。每个组的乘积项被展示了出来,其结果就是最小乘积项之和表达式:

$$\overline{D} + \overline{B}C$$

记住这个最小表达式等价于原来的标准表达式。

相关问题:使用卡诺图来简化下面的乘积项之和表达式:

$$\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}YZ + W\overline{X}\,\overline{Y}Z + \overline{W}YZ + W\overline{X}\,\overline{Y}\,\overline{Z}$$

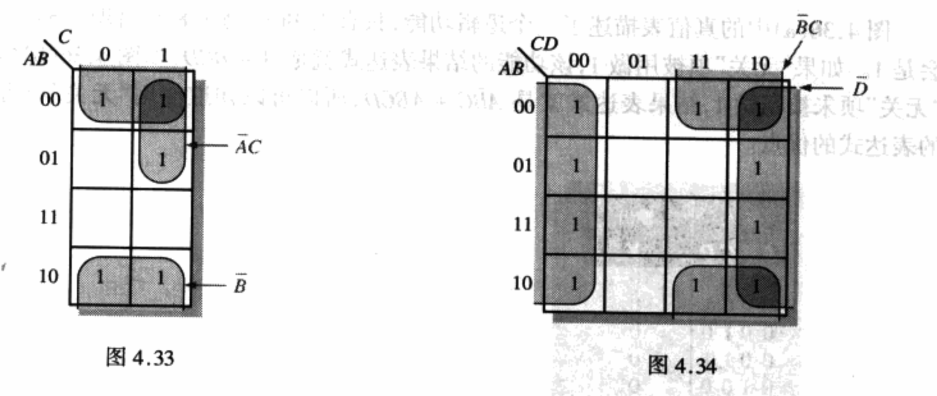


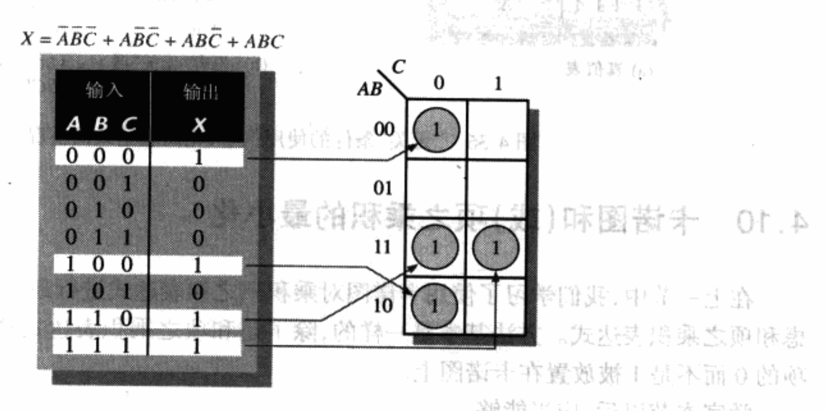
图 4.33

图 4.34

4.9.4 直接从真值表映射

我们已经看到了怎样映射布尔表达式，现在将学习怎样从真值表直接得到卡诺图。回顾一下，真值表给出了一个布尔表达式所有可能输入变量组合所对应的输出。布尔表达式的一个例子及它的真值表表示法如图 4.35 所示。注意在这个真值表中，对于 4 个不同的输入变量组合输出 X 为 1。真值表输出列中的 1 被直接映射到卡诺图的小方格中，这些小方格对应于相关的输入变量组合的值，如图 4.35 所示。在这个图表中，可以看到布尔表达、真值表及卡诺图仅仅是表示逻辑功能的不同方式而已。

$X = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + AB\bar{C} + ABC$



输入 A B C	输出 X
0 0 0	1
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	1

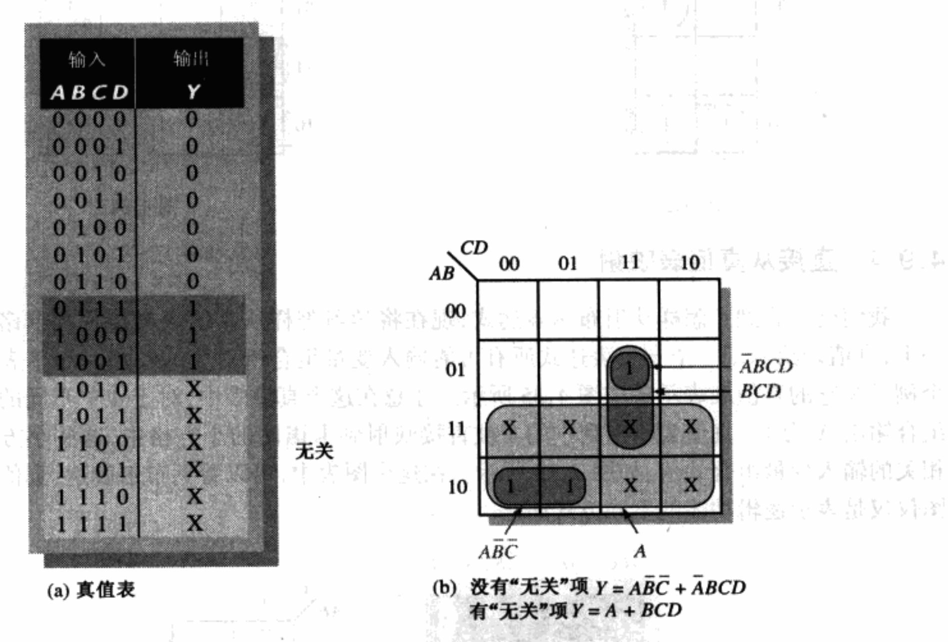
图 4.35　真值表到卡诺图的直接映射

4.9.5 “无关”条件

有时候会发生一种情况，其中有些输入变量组合是不被允许的。例如，记得在第 2 章中所介绍的 BCD 代码，有 6 个无效的组合：1010、1011、1100、1101、1110 及 1111。因为这些不允许的状态在涉及 BCD 码的应用中永远不会发生，它们可以被当做“无关”项来处理，因为它们对输出没有影响。也就是说，对于这些“无关”项，可以赋予输出一个 1 或者 0，但是它们永远不会发生。

在卡诺图中“无关”项的使用有其好处。图 4.36 中在“无关”项的小方格处放置一个 X。在对 1 分组时，X 可以作为 1 以获取更大的组，或者被当做 0，如果使用方便。组越大，那么结果得到的乘积项也越简单。

图 4.36(a)中的真值表描述了一个逻辑功能,只有当 BCD 码 7、8、9 出现在输上时,输出才会是 1。如果"无关"项被用做 1,该功能的结果表达式就是 $A + BCD$,如图 4.36(b)所示。如果"无关"项未被用做 1,结果表达式就是 $A\bar{B}\bar{C} + \bar{A}BCD$,所以可以知道使用"无关"项得到最简单的表达式的优点。



(a) 真值表

(b) 没有"无关"项 $Y = A\bar{B}\bar{C} + \bar{A}BCD$
有"无关"项 $Y = A + BCD$

图 4.36 "无关"条件的使用例子,化简表达式的情况

4.10 卡诺图和(或)项之乘积的最小化

在上一节中,我们学习了使用卡诺图对乘积项之和表达式进行最小化。在本节,将重点考虑和项之乘积表达式。方法基本是一样的,除了是和项之乘积表达式这点不同。表示标准和项的 0 而不是 1 被放置在卡诺图上。

学完本节以后,应当能够

- 把最大项之乘积表达式映射到卡诺图上
- 把图上的 0 组合为最大小组
- 为图上的每一个组确定最小和项
- 组合最小和项以形成最小和项之乘积表达式
- 使用卡诺图在和项之乘积及乘积项之和之间进行转换

4.10.1 映射最大项之乘积表达式

对于最大项之乘积表达式来说,为表达式中的每一个和项在卡诺图上放置一个 0。每一个 0 都被放置在与和项数值相对应的小方格上。例如,对于和项 $A + \bar{B} + C$,一个 0 就会放入 3 变量图上的 010 单元。

当一个和项之乘积表达式被完全映射后,在卡诺图上将会有一些 0,它们的数目等于最大项之积表达式中和项的数目。不具有 0 的小方格对应于表达式为 1 的情况。通常情况下,当处理和项之乘积表达式时,1 会被省略。下面的步骤及图 4.37 中的阐释展示了映射过程。

步骤 1: 确定最大项之积表达式中每一个和项的二进制数值。这个数值使得该项等于 0。

步骤 2: 每一个和项被计算出来后,就在卡诺图相应的小方格上放置一个 0。

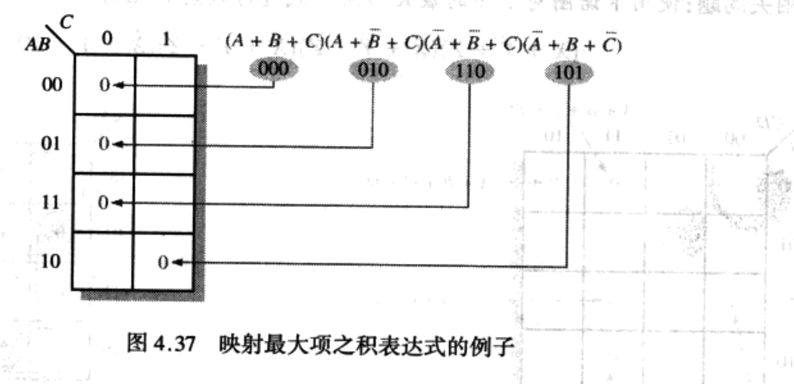


图 4.37　映射最大项之积表达式的例子

例 4.30　把下面的最大项之积表达式映射到卡诺图上:

$$(\overline{A}+\overline{B}+C+D)(\overline{A}+B+\overline{C}+\overline{D})(A+B+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})(A+B+\overline{C}+\overline{D})$$

解: 该表达式的计算如下所示,为表达式中的每一个最大项,都在图 4.38 中的 4 变量卡诺图中放置一个 0。

$$\underset{1100}{(\overline{A}+\overline{B}+C+D)}\underset{1011}{(\overline{A}+B+\overline{C}+\overline{D})}\underset{0010}{(A+B+\overline{C}+D)}\underset{1111}{(\overline{A}+\overline{B}+\overline{C}+\overline{D})}\underset{0011}{(A+B+\overline{C}+\overline{D})}$$

相关问题: 将下面的最大项之积表达式映射到卡诺图上:

$$(A+\overline{B}+\overline{C}+D)(A+B+C+\overline{D})(A+B+C+D)(\overline{A}+B+\overline{C}+D)$$

4.10.2　和项之乘积表达式的卡诺图化简

和项之乘积表达式的最小化过程和乘积项之和表达式的最小化过程基本上是一样的,只不过先对 0 分组而产生最小和项,而不是将 1 分组而产生最小和项。对 0 分组的法则和对 1 分组的法则是一样的,这在 4.9 节中已经学习过了。

例 4.31　使用卡诺图最小化下面的最大项之积表达式:

$$(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)$$

解: 该表达式的二进制数值组合是

$$(0+0+0)(0+0+1)(0+1+0)(0+1+1)(1+1+0)$$

对最大项之积表达式进行映射,然后对小方格分组,如图 4.39 所示。

注意:利用 4 个小方格组中的 0,考虑 110 小方格中的 0 是如何被两个小方格组包含进去的。每组的和项在图上标出,最后得到的最小和项之乘积表达式是

$$A(\overline{B} + C)$$

记住最小和项之乘积表达式和原始的最大项表达式等价。

由灰色区域给出的对 1 分组产生的乘积项之和表达式和对 0 分组是等价的:

$$AC + A\overline{B} = A(\overline{B} + C)$$

相关问题:使用卡诺图对下面的最大项表达式进行映射和化简:

$$(X + \overline{Y} + Z)(X + \overline{Y} + \overline{Z})(\overline{X} + \overline{Y} + Z)(\overline{X} + Y + Z)$$

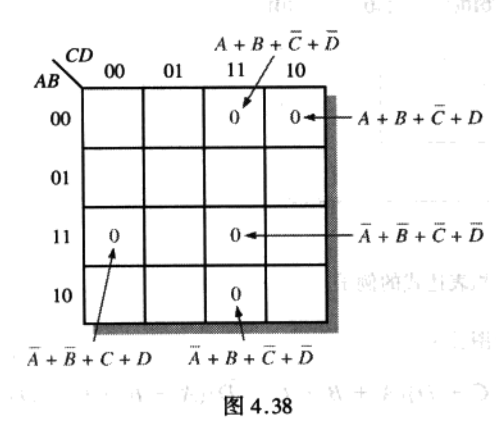


图 4.38

图 4.39

例 4.32 使用卡诺图对下面的和项之乘积表达式进行映射和化简。

$$(B + C + D)(A + B + \overline{C} + D)(\overline{A} + B + C + \overline{D})(A + \overline{B} + C + D)(\overline{A} + \overline{B} + C + D)$$

解:第一项必须扩展为 $\overline{A} + B + C + D$ 和 $A + B + C + D$ 以得到最大项之积表达式,然后再将其映射;小方格如图 4.40 所示进行分组。每个组所对应的和项及最小和项之乘积表达式为

$$(C + D)(A + B + D)(\overline{A} + B + C)$$

记住这个最小和项之乘积表达式等价于原始的最大项之积表达式。

相关问题:使用卡诺图对下面的 POS 表达式进行简化:

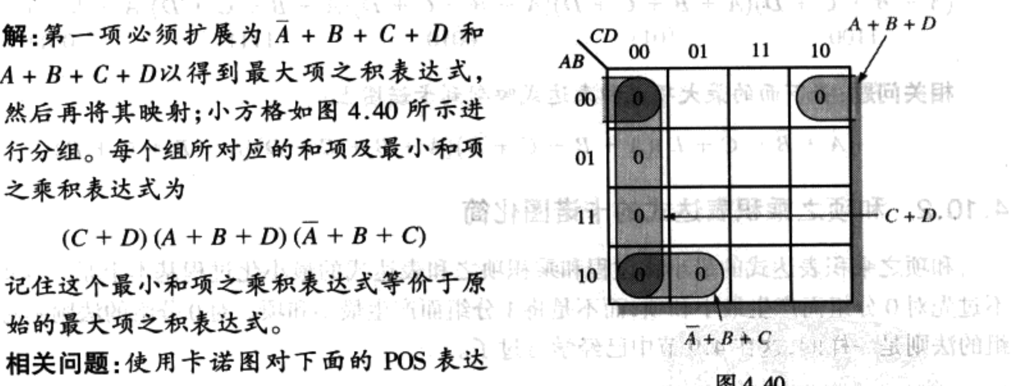


图 4.40

$$(W + \overline{X} + Y + \overline{Z})(W + X + Y + Z)(W + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + Z)$$

4.10.3 使用卡诺图在和项之乘积和乘积项之和之间转换

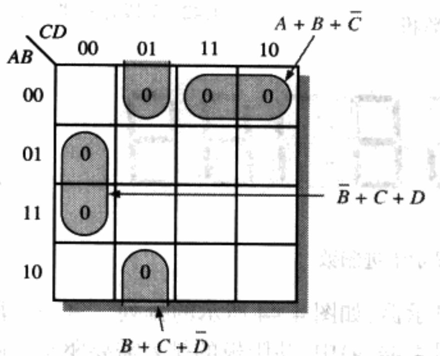
当和项之乘积表达式被映射后,就可以很容易地直接从卡诺图将其转换为等价的乘积项之和的形式。同样,给出一个映射的乘积项之和表达式,与其等价的和项之乘积表达式也可以直接从卡诺图推导出。这就提供了比较这两种最小表达式形式的一种好方法,以确定其中一种形式是否可以比另一种形式使用更少的门来实现。

对于一个和项之乘积表达式,所有不包含 0 的小方格都包含 1,从这里我们就可以导出乘积项之和表达式。类似地,对于一个乘积项之和表达式,所有不包含 1 的表达式都包含 0,从这里可以导出和项之乘积表达式。例 4.33 阐述了这个变换。

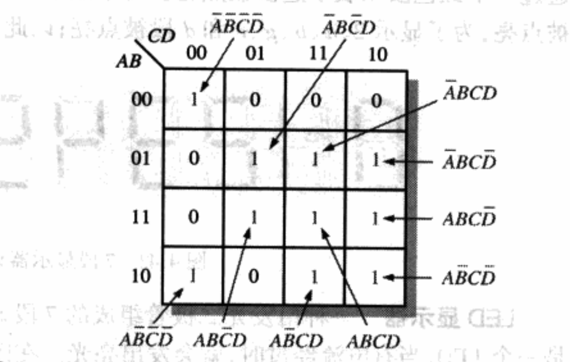
例 4.33　使用卡诺图将下面的最大项之和表达式变换为最小的和项之乘积表达式、最小项表达式及最小乘积项之和表达式。

$$(\overline{A} + \overline{B} + C + D)(A + \overline{B} + C + D)(A + B + C + \overline{D})(A + B + \overline{C} + \overline{D})(\overline{A} + B + C + \overline{D})(A + B + \overline{C} + D)$$

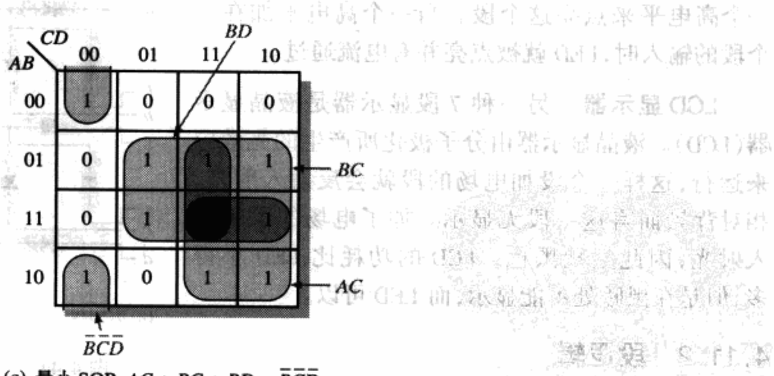
解:这个最大项表达式的 0 被映射并被分组以得到图 4.41(a)所示的最小和项之乘积表达式。在图 4.41(b)中,在不含有 0 的小方格中添加 1。从每个包含 1 的小方格中,就会得到所示的最小项。这些乘积项形成了最小项之和表达式。在图 4.41(c)中,对 1 进行了分组,并且得到了最小乘积项之和表达式。



(a) 最小 POS: $(A + B + C)(\overline{B} + \overline{C} + \overline{D})(B + C + \overline{D})\ + \overline{D})$



(b) 标准 SOP:
$\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D} +$
$A\overline{B}\overline{C}\overline{D} + AB\overline{C}D + A\overline{B}CD + ABCD$



(c) 最小 SOP: $AC + BC + BD + \overline{B}\overline{C}\overline{D}$

图 4.41

相关问题:使用卡诺图将下面的表达式变换为最小和项之乘积形式:

$$(W + \overline{X} + Y + \overline{Z})(\overline{W} + X + \overline{Y} + \overline{Z})(\overline{W} + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + \overline{Z})$$

4.11 数字系统应用

7段显示器被应用于许多种产品中。第1章中描述的药片计数和控制系统具有7段显示器。这些显示器和逻辑电路一起使用,逻辑电路对二-十进制数(BCD)译码,并在显示器上点亮相应的数字。在这个数字系统应用中,将集中于最小门的设计,阐明布尔表达式和卡诺图的应用。

4.11.1 7段显示器

图4.42展示了一个普通的显示格式,由7个元件或者段组成。点亮这些段的一些组合,可以使得每个十进制数都能够显示出来。图4.43展示了这种用于每一个十进制数的显示方法,通过一个红色段以表示这段被点亮。为了显示1,b段和c段将被点亮,为了显示2、a、b、g、e和d段被点亮;以此类推。

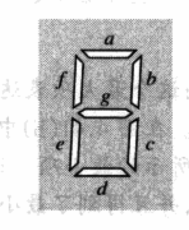


图4.42 7段显示器格式

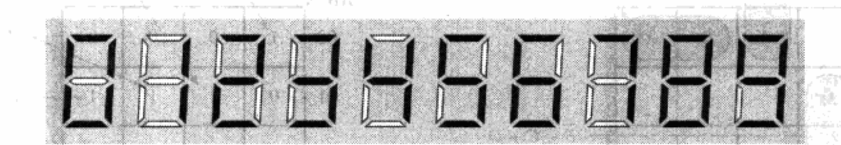


图4.43 7段显示器显示十进制数

LED显示器 一种由发光二极管组成的7段显示器,如图4.44所示的排列。每一个段都是一个LED,当有电流经过时,就会发出亮光。在图4.44(a)中,共阳极的排列需要驱动电路提供一个低电平来点亮这个段。当一个低电平加在一个段的输入时,LED就被点亮并有电流通过。在图4.44(b)中,共阴极的排列需要驱动电路提供一个高电平来点亮这个段。当一个高电平加在一个段的输入时,LED就被点亮并有电流通过。

LCD显示器 另一种7段显示器是液晶显示器(LCD)。液晶显示器由分子极化所产生的光效应来运行,这样一个没加电场的段就会反射入射光,相对背景而言这一段无显示。加了电场的段吸收入射光,因此呈现黑色。LCD的功耗比LED小得多,但是在黑暗处不能显示,而LED可以。

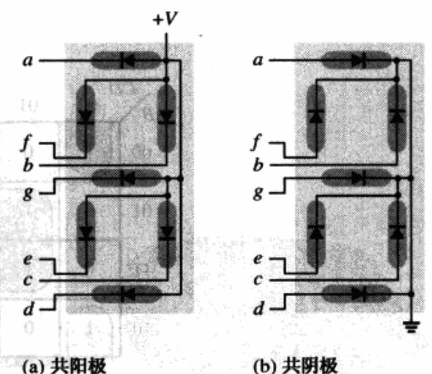


图4.44 7段LED显示器的结构

4.11.2 段逻辑

每一个段都被用于不同的十进制数字,但是没有一个段应用于所有的10个数字。所以,每一个段都必须利用它自己的译码电路来驱动,这个电路可以检测任何数字需要使用的段。在图4.42和图4.43中,把每一个显示数字所需要驱动的段都确定下来并列在表4.9中。

表 4.9　每个十进制数字的有效段

数字	点亮的段
0	*a, b, c, d, e, f*
1	*b, c*
2	*a, b, d, e, g*
3	*a, b, c, d, g*
4	*b, c, f, g*
5	*a, c, d, f, g*
6	*a, c, d, e, f, g*
7	*a, b, c*
8	*a, b, c, d, e, f, g*
9	*a, b, c, d, f, g*

4.11.3　段逻辑的真值表

段译码逻辑需要 4 个二－十进制数(BCD)输入及 7 个输出，每一个都对应于显示器中的一个段，如图 4.45 中所示的方块图。展示在表 4.10 中的多输出真值表，实际上就是 7 个真值表的集合，并且可以将每个段分成相应的独立表。表中段输出列的 1 表示有效状态下的段。

由于 BCD 代码并不包含二进制数值 1010、1011、1100、1101、1110 和 1111，这些组合绝对不会出现在输入上，因此可以把它们看做“无关”(×)条件，如同在真值表中所指示的一样。为了和大多数 IC 生产商的实际产品保持一致，在这个特殊的应用中，*A* 表示最低有效位而 *D* 表示最高有效位。

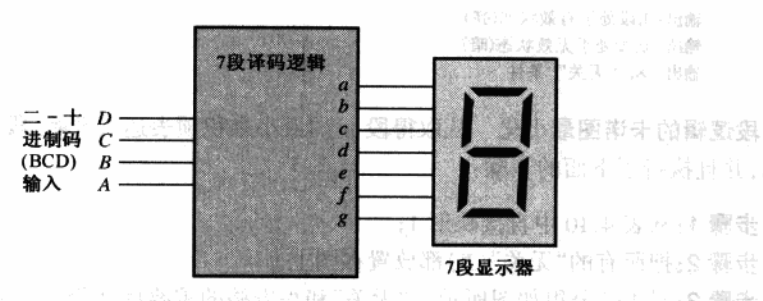


图 4.45　7 段逻辑和显示的方块图

段逻辑的布尔表达式　从真值表中，可以为每一个段写出最小项之和或者最大项之积表达式。例如，段 *a* 的最小项之和表达式为

$$a = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}\,\overline{C}BA + \overline{D}C\overline{B}A + \overline{D}CB\overline{A} + \overline{D}CBA + D\overline{C}\,\overline{B}\,\overline{A} + D\overline{C}\,\overline{B}A$$

而段 *e* 的最小项之和表达式为

$$e = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}CB\overline{A} + D\overline{C}\,\overline{B}\,\overline{A}$$

其他段的表达式可以类似地开发出来。正如所见，段 *a* 的表达式具有 8 个乘积项并且段 *e* 的表达式具有 4 个乘积项，用以表示每一个驱动此段的 *BCD* 输入。这就说明段 *a* 逻辑的最小项之和的实现需要一个与－或电路，这个电路由 8 个四输入与门和一个八输入或门。段 *e*

逻辑的实现需要四个四输入与门及一个四输入或门。在这两个例子中,都需要 4 个反相器来产生每个变量的反码。

表 4.10　7 段逻辑的真值表

十进制数	输入				段输出						
	D	*C*	*B*	*A*	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

输出 = 1,段处于有效状态(亮)
输出 = 0,段处于无效状态(暗)
输出 = ×,“无关”条件

段逻辑的卡诺图最小化　从取得段 *a* 的最小乘积项表达式开始,段 *a* 的卡诺图如图 4.46 所示,并且执行了下面的步骤:

步骤 1: 从表 4.10 中直接映射 1;

步骤 2: 把所有的“无关”(X)都放置在图上。

步骤 3: 对 1 的分组如图所示。“无关”和小方格的重叠用于形成尽可能大的组。

步骤 4: 为每一个组写出最小乘积项,并且将这些项加起来形成最小乘积项之和表达式。

记住“无关”并不一定包含在组内,但是在以上情况中,它们都被用到了。同样,注意利用角上小方格的“环绕”相邻,角上小方格中的 1 和“无关”(X)成为一组。

段 *a* 逻辑的最小实现　图 4.47 中的卡诺图中取得的段 *a* 逻辑的最小乘积项之和表达式是

$$D + B + CA + \overline{C}\,\overline{A}$$

这个表达式可以用两个二输入与门、一个四输入或门及两个反相器来实现,如图 4.47 所示。把这个电路和前面讨论的段 *a* 逻辑最小项之和的实现相比较,就会发现门和反相器的数目已经从 13 个减少到了 5 个。作为结果,互相连接的数目也被大幅度减少。

其余 6 个段的最小逻辑(*b*、*c*、*d*、*e*、*f* 和 *g*)可以用相似的方法得到。

最小项之和表达式:

$\overline{D}\overline{C}\overline{B}\overline{A} + \overline{D}\overline{C}B\overline{A} + \overline{D}\overline{C}BA + \overline{D}C\overline{B}A + \overline{D}CB\overline{A} + \overline{D}CBA + D\overline{C}\overline{B}\overline{A} + D\overline{C}B\overline{A}$

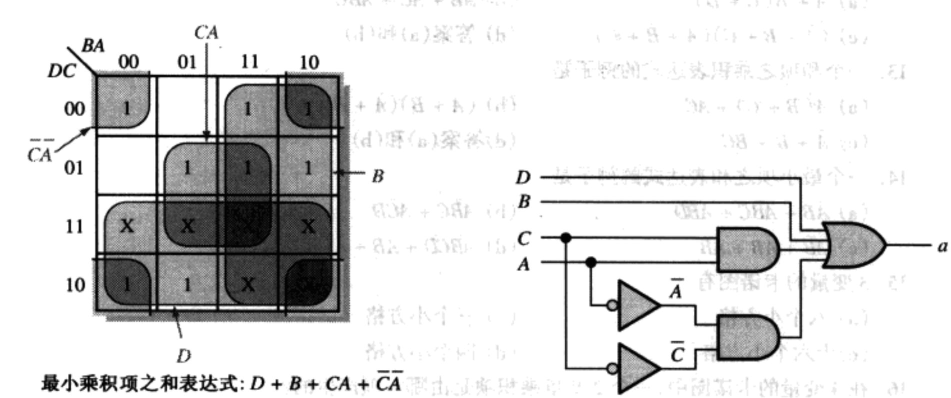


图 4.46 段 a 逻辑表达式的卡诺图最小化

图 4.47 7 段显示器的段 a 的最小逻辑实现

自测题 (答案在本章的结尾)

1. 变量的反码始终是
   (a) 0　(b) 1　(c) 等于变量　(d) 变量的反相
2. 布尔表达式 $A + \overline{B} + C$ 是
   (a) 一个和项　(b) 一个文字项　(c) 一个乘积项　(d) 一个补码项
3. 布尔表达式 $A\overline{B}C\overline{D}$ 是
   (a) 一个和项　(b) 一个乘积项　(c) 一个文字项　(d) 始终是 1
4. 表达式 $A\overline{B}CD + A\overline{B} + \overline{C}D + B$ 的域是
   (a) A 和 D　(b) 只是 B　(c) A、B、C 和 D　(d) 都不是
5. 按照加法的交换律:
   (a) $AB = BA$　(b) $A = A + A$
   (c) $A + (B + C) = (A + B) + C$　(d) $A + B = B + A$
6. 按照乘法的结合律:
   (a) $B = BB$　(b) $A(BC) = (AB)C$　(c) $A + B = B + A$　(d) $B + B(B + 0)$
7. 按照乘法的分配律:
   (a) $A(B + C) = AB + AC$　(b) $A(BC) = ABC$
   (c) $A(A + 1) = A$　(d) $A + AB = A$
8. 下面哪一个不是布尔代数的有效法则?
   (a) $A + 1 = 1$　(b) $A = \overline{A}$　(c) $A \cdot A = A$　(d) $A + 0 = A$
9. 下面哪一个法则表述了如果与门的一个输入总是 1,那么输出就等于另外一个输入?
   (a) $A + 1 = 1$　(b) $A + A = A$　(c) $A \cdot A = A$　(d) $A \cdot 1 = A$
10. 按照狄摩根定理,下面的等式是正确的有
    (a) $\overline{AB} = \overline{A} + \overline{B}$　(b) $\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$
    (c) $\overline{A + B + C} = \overline{A}\overline{B}\overline{C}$　(d)所以这些等式
11. 布尔表达式 $X = AB + CD$ 表示
    (a) 两个或运算再相与　(b) 一个四输入与门
    (c) 两个与运算再相或　(d) 一个异或门

12. 一个乘积项之和表达式的例子是

(a) $A+B(C+D)$　　(b) $\bar{A}B+A\bar{C}+A\bar{B}C$

(c) $(\bar{A}+B+C)(A+\bar{B}+C)$　　(d) 答案(a)和(b)

13. 一个和项之乘积表达式的例子是

(a) $A(B+C)+A\bar{C}$　　(b) $(A+B)(\bar{A}+B+\bar{C})$

(c) $\bar{A}+\bar{B}+BC$　　(d)答案(a)和(b)

14. 一个最小项之和表达式的例子是

(a) $\bar{A}B+A\bar{B}C+AB\bar{D}$　　(b) $A\bar{B}C+A\bar{C}D$

(c) $A\bar{B}+\bar{A}B+AB$　　(d) $A\bar{B}C\bar{D}+\bar{A}B+\bar{A}$

15. 3 变量的卡诺图有

(a) 八个小方格　　(b) 三个小方格

(c)十六个小方格　　(d) 四个小方格

16. 在 4 变量的卡诺图中,一个 2 变量乘积项是由哪一项产生的?

(a) 1 的 2 个小方格组　　(b) 1 的 8 个小方格组

(c) 1 的 4 个小方格组　　(d) 0 的 4 个小方格组

17. 在卡诺图中, 把 0 圈在一起会产生

(a) 一个和项之乘积表达式　　(b) 积之和表达式　　(c) 一个“无关”条件　　(d) 与－或逻辑

习题

4.1 节　布尔运算和表达式

1. 使用布尔符号,写出一个表达式,无论何时它的一个或多个变量(A,B,C,D)为 1 时, 这个表达式为 1。
2. 写出一个表达式, 当且仅当所有的变量(A,B,C,D,E)为 1 时, 这个表达式才是 1。
3. 写出一个表达式,当一个或多个变量(A,B,C)为 1 时, 这个表达式才是 0。
4. 计算下面的运算:

(a) $0+0+1$　　(b) $1+1+1$　　(c) $1\cdot 0\cdot 0$

(d) $1\cdot 1\cdot 1$　　(e) $1\cdot 0\cdot 1$　　(f) $1\cdot 1+0\cdot 1\cdot 1$

5. 求使得乘积项为 1,和项为 0 的变量的值:

(a) AB　　(b) $A\bar{B}C$　　(c) $A+B$　　(d) $\bar{A}+B+\bar{C}$

(e) $\bar{A}+\bar{B}+C$　　(f) $\bar{A}+B$　　(g) $A\bar{B}\bar{C}$

6. 对于变量的所有可能的值,求 X 的值。

(a) $X=(A+B)C+B$　　(b) $X=\overline{(A+B)}C$　　(c) $X=A\bar{B}C+AB$

(d) $X=(A+B)(\bar{A}+B)$　　(e) $X=(A+BC)(\bar{B}+\bar{C})$

4.2 节　布尔代数的定理和法则

7. 基于下面的每一个等式,指出布尔代数的定律:

(a) $A\bar{B}+CD+A\bar{C}D+B=B+A\bar{B}+A\bar{C}D+CD$

(b) $AB\bar{C}D+\overline{ABC}=D\bar{C}BA+\overline{CBA}$

(c) $AB(CD+E\bar{F}+GH)=ABCD+ABE\bar{F}+ABGH$

8. 基于下面的每一个等式,指出布尔代数的定律:

(a) $\overline{\overline{AB+CD}}+\overline{EF}=AB+CD+\overline{EF}$　　(b) $A\bar{A}B+AB\bar{C}+AB\bar{B}=AB\bar{C}$

(c) $A(BC+BC)+AC=A(BC)+AC$　　(d) $AB(C+\bar{C})+AC=AB+AC$

(e) $A\bar{B}+A\bar{B}C=A\bar{B}$　　(f) $ABC+\overline{AB}+\overline{ABC}D=ABC+\overline{AB}+D$

4.3 节　狄摩根定理

9. 对下面的每一个表达式,应用狄摩根定理:

(a) $\overline{A+\overline{B}}$　(b) $\overline{A}B$　(c) $\overline{A+B+C}$　(d) $\overline{ABC}$

(e) $\overline{A(B+C)}$　(f) $\overline{AB}+\overline{CD}$　(g) $\overline{AB+CD}$　(h) $\overline{(A+\overline{B})(\overline{C}+D)}$

10. 对下面的每一个表达式,应用狄摩根定理:

(a) $\overline{A\overline{B}(C+\overline{D})}$　(b) $\overline{AB(CD+EF)}$

(c) $\overline{(A+\overline{B}+C+\overline{D})}+\overline{ABC\overline{D}}$　(d) $\overline{\overline{(\overline{A}+B+C+D)}\;\overline{(A\overline{B}\overline{C}D)}}$

(e) $\overline{\overline{AB}(CD+\overline{E}F)(\overline{AB}+\overline{CD})}$

11. 对下面的每一个表达式,应用狄摩根定理:

(a) $\overline{\overline{(\overline{ABC})(\overline{EFG})}+\overline{(\overline{HIJ})(\overline{KLM})}}$　(b) $\overline{(A+\overline{B\overline{C}}+CD)}+\overline{\overline{BC}}$　(c) $\overline{\overline{(\overline{A+B})(\overline{C+D})(\overline{E+F})(\overline{G+H})}}$

4.4 节　逻辑电路的布尔分析

12. 在图 4.48 中,为每一个逻辑门写出布尔表达式。

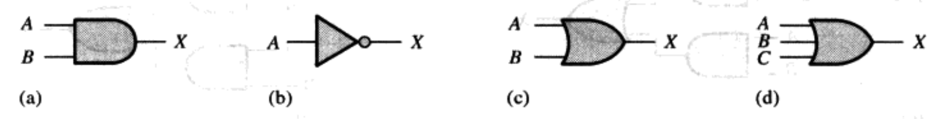


图 4.48

13. 在图 4.49 中,为每一个逻辑门写出布尔表达式。

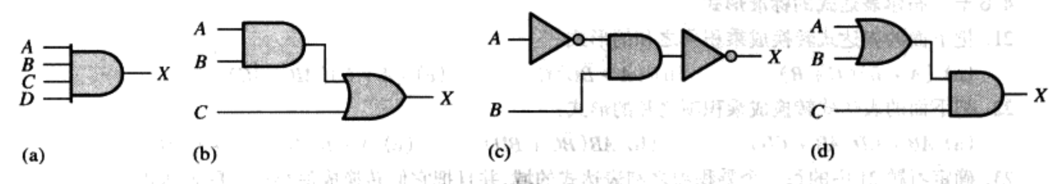


图 4.49

14. 绘制下面每一个表达式的逻辑电路:

(a) $A+B+C$　(b) ABC　(c) $AB+C$　(d) $AB+CD$

15. 绘制下面每一个表达式的逻辑电路:

(a) $A\overline{B}+\overline{A}B$　(b) $AB+\overline{A}\overline{B}+\overline{A}BC$

(c) $\overline{A}B(C+\overline{D})$　(d) $A+B[C+D(B+\overline{C})]$

16. 为下面的每一个布尔表达式构建真值表:

(a) $A+B$　(b) AB　(c) $AB+BC$

(d) $(A+B)C$　(e) $(A+B)(\overline{B}+C)$

4.5 节　用布尔代数进行化简

17. 使用布尔代数技术,尽可能地简化下面的表达式:

(a) $A(A+B)$　(b) $A(\overline{A}+AB)$　(c) $BC+\overline{B}C$

(d) $A(A+\overline{A}B)$　(e) $A\overline{B}C+\overline{A}BC+\overline{A}\overline{B}C$

18. 使用布尔代数,简化下面的表达式:

(a) $(A+\overline{B})(A+C)$　(b) $\overline{A}B+\overline{A}B\overline{C}+\overline{A}BCD+\overline{A}B\overline{C}\overline{D}E$　(c) $AB+\overline{\overline{A}B}C+A$

(d) $(A+\overline{A})(AB+AB\overline{C})$　(e) $AB+(\overline{A}+\overline{B})C+AB$

19. 使用布尔代数,简化下面的表达式:

(a) $BD+B(D+E)+\overline{D}(D+F)$　(b) $\overline{A}\overline{B}C+\overline{(A+B+\overline{C})}+\overline{A}\overline{B}C\overline{D}$

(c) $(B+BC)(B+\bar{B}C)(B+D)$ (d) $ABCD+AB(\overline{CD})+(\overline{AB})CD$

(e) $ABC[AB+\bar{C}(BC+AC)]$

20. 确定图 4.50 中的哪些电路是等价的？

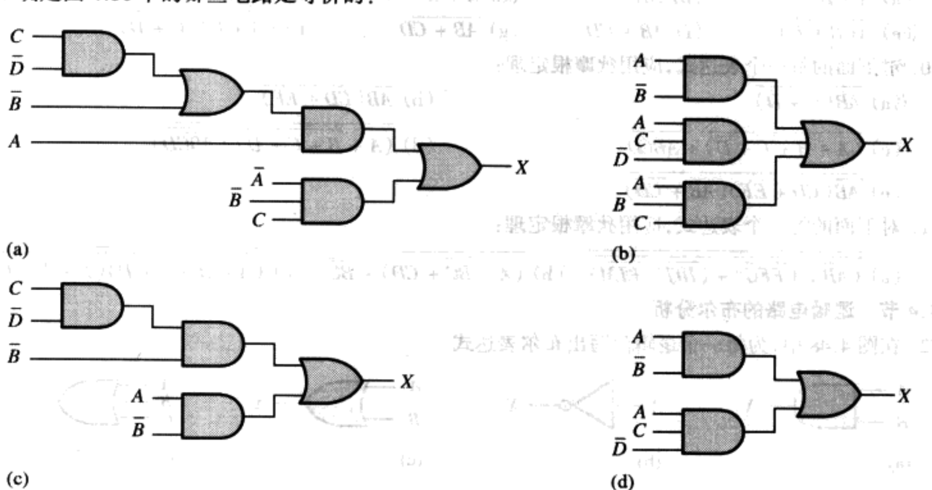


图 4.50

4.6 节 布尔表达式的标准形式

21. 把下面的表达式转换成乘积项之和的形式：

(a) $(A+B)(C+\bar{B})$ (b) $(A+\bar{B}C)C$ (c) $(A+C)(AB+AC)$

22. 把下面的表达式转换成乘积项之和的形式：

(a) $AB+CD(A\bar{B}+CD)$ (b) $AB(\bar{B}\bar{C}+BD)$ (c) $A+B[AC+(B+\bar{C})D]$

23. 确定习题 21 中的每一个乘积项之和表达式的域，并且把它们转换成最小项之和表达式。

24. 把习题 22 中的乘积项之和表达式转换成最小项之和表达式。

25. 确定习题 23 的最小项之和表达式中的每一个乘积项的二进制值。

26. 确定习题 24 的最小项之和表达式中的每一个乘积项的二进制值。

27. 把习题 23 中的每一个最小项之和表达式转换成最大项之积表达式。

28. 把习题 24 中的每一个最小项之和表达式转换成最大项之积表达式。

4.7 节 布尔表达式和真值表

29. 为下面每一个最小项之和表达式开发真值表：

(a) $A\bar{B}C+\bar{A}B\bar{C}+ABC$ (b) $\bar{X}\bar{Y}\bar{Z}+\bar{X}\bar{Y}Z+XY\bar{Z}+X\bar{Y}Z+\bar{X}YZ$

30. 为下面每一个最小项之和表达式开发真值表：

(a) $\bar{A}B\bar{C}D+\bar{A}BC\bar{D}+A\bar{B}\bar{C}D+\bar{A}\bar{B}\bar{C}\bar{D}$

(b) $WXYZ+WXY\bar{Z}+\bar{W}XYZ+W\bar{X}YZ+WX\bar{Y}Z$

31. 为下面每一个乘积项之和表达式开发真值表：

(a) $\bar{A}B+AB\bar{C}+\bar{A}\bar{C}+A\bar{B}C$ (b) $\bar{X}+Y\bar{Z}+WZ+X\bar{Y}Z$

32. 为下面每一个最大项之积表达式开发真值表：

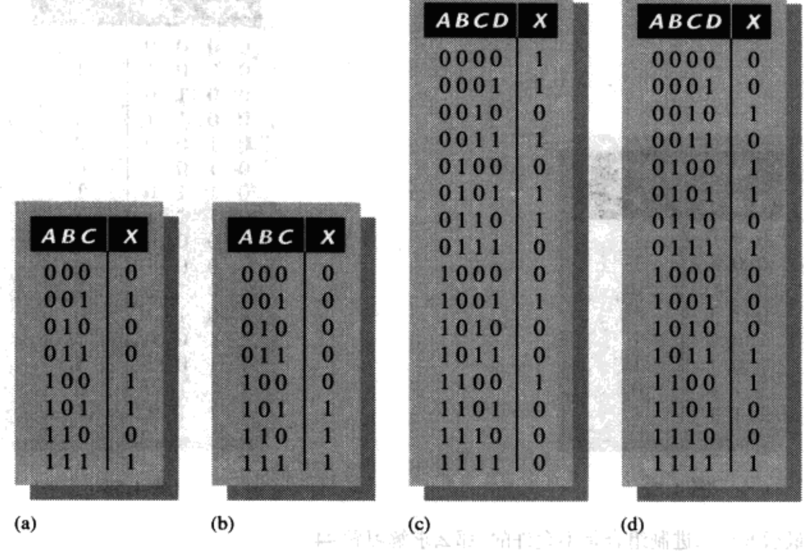
(a) $(\bar{A}+\bar{B}+\bar{C})(A+B+C)(A+\bar{B}+C)$

(b) $(\bar{A}+B+\bar{C}+D)(A+\bar{B}+C+\bar{D})(A+\bar{B}+\bar{C}+D)(\bar{A}+B+C+\bar{D})$

33. 为下面每一个最大项之积表达式开发真值表：

(a) $(A+B)(A+C)(A+B+C)$ (b) $(A+\bar{B})(A+\bar{B}+\bar{C})(B+C+\bar{D})(\bar{A}+B+\bar{C}+D)$

34. 为图 4.51 中的每一个真值表,推导出一个最小项之和表达式和最大项之积表达式。



(a)

ABC	X
000	0
001	1
010	0
011	0
100	1
101	1
110	0
111	1

(b)

ABC	X
000	0
001	0
010	0
011	0
100	0
101	1
110	1
111	1

(c)

ABCD	X
0000	1
0001	1
0010	0
0011	1
0100	0
0101	1
0110	1
0111	0
1000	0
1001	1
1010	0
1011	0
1100	1
1101	0
1110	0
1111	0

(d)

ABCD	X
0000	0
0001	0
0010	1
0011	0
0100	1
0101	1
0110	0
0111	1
1000	0
1001	0
1010	0
1011	1
1100	1
1101	0
1110	0
1111	1

图 4.51

4.8 节　卡诺图

35. 绘制一个 3 变量的卡诺图,并根据二进制值标记每一个小方格。

36. 绘制一个 4 变量的卡诺图,并根据二进制值标记每一个小方格。

37. 在一个 3 变量的卡诺图中,写出每个小方格的最小项。

4.9 节　卡诺图乘积项之和的最小化

38. 使用卡诺图为每一个表达式求出最小乘积项之和的形式:

　(a) $\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C$　　(b) $AC(\bar{B} + C)$

　(c) $\bar{A}(BC + B\bar{C}) + A(BC + B\bar{C})$　　(d) $\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C}$

39. 使用卡诺图简化为每一个表达式,以得到最小乘积项之和的形式:

　(a) $\bar{A}\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC + AB\bar{C}$　　(b) $AC[\bar{B} + B(B + \bar{C})]$

　(c) $DE\bar{F} + \bar{D}E\bar{F} + \bar{D}\bar{E}\bar{F}$

40. 把表达式扩展成为最小项之和的形式:

　(a) $AB + A\bar{B}C + ABC$　　(b) $A + BC$

　(c) $A\bar{B}\bar{C}D + AC\bar{D} + B\bar{C}D + \bar{A}BC\bar{D}$　　(d) $A\bar{B} + A\bar{B}\bar{C}D + CD + B\bar{C}D + ABCD$

41. 用卡诺图对习题 40 中的每个表达式进行最小化。

42. 使用卡诺图把每个表达式简化为最小乘积项之和形式:

　(a) $A + B\bar{C} + CD$

　(b) $\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + ABCD + ABC\bar{D}$

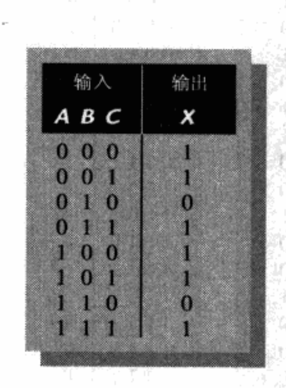
　(c) $\bar{A}B(\bar{C}\bar{D} + \bar{C}D) + AB(\bar{C}\bar{D} + \bar{C}D) + A\bar{B}\bar{C}D$

　(d) $(\bar{A}\bar{B} + A\bar{B})(CD + C\bar{D})$

　(e) $\bar{A}\bar{B} + A\bar{B} + \bar{C}\bar{D} + C\bar{D}$

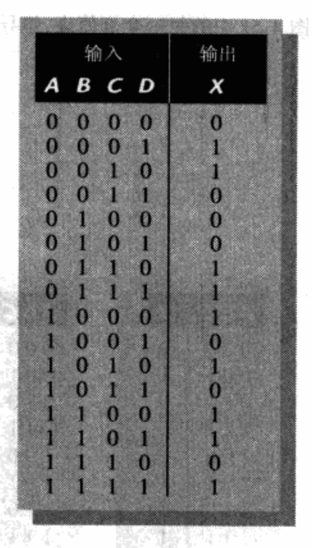
43. 使用卡诺图把图 4.52 中真值表所确定的逻辑函数简化到最小的乘积项之和形式。

44. 使用卡诺图方法把图 4.53 中真值表所确定的逻辑函数的简化为最小的乘积项之和形式



输入 A B C	输出 X
0 0 0	1
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	1

图 4.52



输入 A B C D	输出 X
0 0 0 0	0
0 0 0 1	1
0 0 1 0	1
0 0 1 1	0
0 1 0 0	0
0 1 0 1	0
0 1 1 0	1
0 1 1 1	1
1 0 0 0	1
1 0 0 1	0
1 0 1 0	1
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	0
1 1 1 1	1

图 4.53

45. 假设最后 6 个二进制组合是不允许的,那么求解习题 44。

4.10 节　卡诺图和(或)项之乘积的最小化

46. 使用卡诺图求每个表达式的最小和项之乘积。
    (a) $(A+B+C)(\bar{A}+\bar{B}+\bar{C})(A+\bar{B}+C)$
    (b) $(X+\bar{Y})(\bar{X}+Z)(X+\bar{Y}+\bar{Z})(\bar{X}+\bar{Y}+Z)$
    (c) $A(B+\bar{C})(\bar{A}+C)(A+\bar{B}+C)(\bar{A}+B+\bar{C})$
47. 使用卡诺图简化每个表达式,使之成为最小和项之乘积形式。
    (a) $(A+\bar{B}+C+\bar{D})(\bar{A}+B+\bar{C}+D)(\bar{A}+\bar{B}+\bar{C}+\bar{D})$
    (b) $(X+\bar{Y})(W+\bar{Z})(\bar{X}+\bar{Y}+\bar{Z})(W+X+Y+Z)$
48. 对于图 4.52 的真值表所确定的函数,使用卡诺图简化为最小和项之乘积表达式。
49. 为图 4.53 的真值表的函数,确定最小和项之乘积表达式。
50. 使用卡诺图把下面的和项之乘积表达式转换为最小乘积项之和表达式。
    (a) $(A+\bar{B})(A+\bar{C})(\bar{A}+\bar{B}+C)$
    (b) $(\bar{A}+B)(\bar{A}+\bar{B}+\bar{C})(B+\bar{C}+D)(A+\bar{B}+C+\bar{D})$

4.11 节　数字系统应用

51. 如果需要选择一个在暗光条件下工作的同一种数值显示器,选择 LED 类型的还是 LCD 类型的 7 段显示器?为什么?
52. 解释为什么在 7 段显示的应用中,二进制代码 1010、1011、1100、1101、1110 和 1111 属于"无关"项?
53. 对于段 b,实现最小乘积项之和表达式所需要的门和反相器,比最小项之和表达式所需要的少多少?
58. 对于段 c 到 g,重复习题 57。

自测题答案

1.(d)　2.(a)　3.(b)　4.(c)　5.(d)　6.(b)　7.(a)　8.(b)　9.(d)　10.(d)　11.(c)　12.(b)　13.(b)
14.(c)　15.(a)　16.(c)　17.(a)

第5章　组合逻辑

章节提纲

5.1　基本组合逻辑电路

在第4章中,学习了乘积之和表达式是由与门(AND)和或门(OR)来实现的,与门对应于每一个乘积项,而或门把所有的乘积项加起来。正如所知,这种乘积之和的实现称为与或(AND-OR)逻辑,并且是实现标准布尔函数的基本形式。在这一节中,将会介绍与或及与或非(AND-OR-Invert);同时还要介绍异或门(XOR)和同或门(XNOR),实际就是与或逻辑的一种形式。

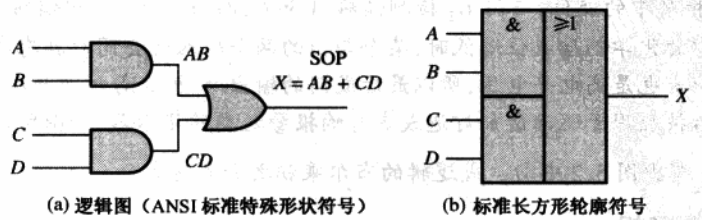
学完本节以后,应当能够

- 分析并应用与-或电路
- 分析并应用与-或-反相电路
- 分析并应用异或门
- 分析并应用同或门

5.1.1　与或逻辑

◇ 与-或逻辑产生一个乘积之和表达式。

图5.1(a)展示了一个与或电路,由两个二输入与门和一个二输入或门组成;图5.1(b)是ANSI标准矩形轮廓符号。与门输出的布尔表达式及输出 X 得到的乘积之和表达式如图所示。一般来说,一个与或电路可以具有任意数目的与门,并且每一个与门都可以具有任意数目的输入。



(a) 逻辑图（ANSI 标准特殊形状符号）　(b) 标准长方形轮廓符号

图5.1　与-或逻辑的一个例子

一个四输入与或逻辑电路的真值表如表5.1所示。中间与门输出(*AB* 和 *CD* 列)也展示在该表中。

表5.1 图5.1中与或逻辑的真值表

输入						输出
A	*B*	*C*	*D*	*AB*	*CD*	*X*
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

与或电路直接实现乘积之和表达式,假设变量的反码(如果有)是可用的。图5.1中与-或电路的运算如下所述:

对于四输入的与或逻辑电路,如果输入 *A* 和输入 *B* 都是高电平(1)或者输入 *C* 和输入 *D* 都是高电平(1)的情况,输出 *X* 就是高电平(1)。

例5.1 在一个化学品加工厂中,一种液体化学物质被应用在了加工过程中。这种化学物质储存在三个不同的储罐中。当任何一个储罐中化学物质的液位降低到某个设定点时,储罐的液位传感器就产生一个高电平电压。

设计一个电路,用以监测每个储罐中的化学物质液位,并指示何时任意两个储罐中的液位降低到设特定点以下。

解:图5.2中的与-或电路具有来自储罐 *A*、*B*、*C* 传感器的输入,如图所示。与门 G_1 检测储罐 *A* 和 *B* 中的液位,与门 G_2 检测储罐 *A* 和 *C*,而与门 G_3 检测储罐 *B* 和 *C*。当任意两个储罐中的化学物质液位过低时,某个与门的两个输入就会同时具有高电平电压,从而使得它的输出也是高电平电压,所以最后或门的输出 *X* 也是高电平电压。这个高电平电压输入随后被应用于驱动诸如灯泡或者音响报警之类的指示器,如图所示。

相关问题:写出图5.2中与-或逻辑的布尔乘积之和表达式。

5.1.2 与或非逻辑

当与或电路的输出被求反(反相)后,就会得到一个与或非电路。回顾一下,与或逻辑直接

实现乘积之和表达式。和之积表达式可以用与或非逻辑来实现，从一个和之积表达式开始，然后开发出相应的与或非表达式：

$$X = (\overline{A} + \overline{B})(\overline{C} + \overline{D}) = (\overline{AB})(\overline{CD}) = \overline{\overline{(\overline{AB})(\overline{CD})}} = \overline{\overline{\overline{AB}} + \overline{\overline{CD}}} = \overline{AB + CD}$$

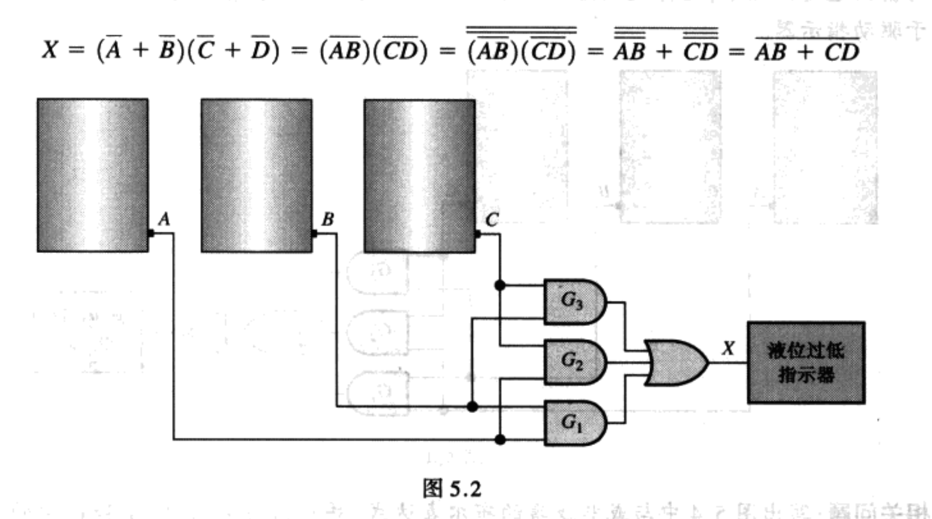


图 5.2

图 5.3(a)中的逻辑图给出了一个与或非电路，并且导出了和之积输出表达式。ANSI 标准的矩形轮廓符号如图 5.3(b)所示。一般来说，一个与或非电路可以具有任意数目的与门，并且每一个与门可以具有任意数目的输入。

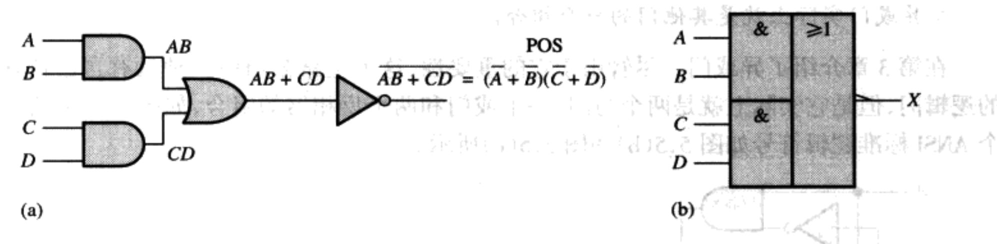


图 5.3　一个产生和之积输出的与或非电路

图 5.3 中的与或非电路的运算可以做如下表述：

对于一个四输入与或非电路，如果输入 *A* 和输入 *B* 都是高电平(1)，或者输入 *C* 和输入 *D* 都是高电平(1)，那么输出 *X* 就是低电平(0)。

从表 5.1 中的与或真值表中可以开发出一个与或非真值表，只需将输出列中的所有的 1 改变为 0，将所有的 0 改变为 1 即可。

例 5.2　例 5.1 中的化学物质储罐的传感器被一个新模型所取代，这个模型在储罐中的液位降低到某个设定点时，就会产生一个低电平电压而不是高电平电压。

修改图 5.2 中的电路以运行不同的输入电平，当任意两个储罐中的液位低于设定点时，仍然产生一个高电平电压输出用以驱动指示器。请给出逻辑图。

解：图 5.4 中的与或非电路具有来自储罐 *A*、*B* 和 *C* 传感器的输入，如图所示。与门 G_1 检测储罐 *A* 和 *B* 中的液位，与门 G_2 检测储罐 *A* 和 *C*，而与门 G_3 检测储罐 *B* 和 *C*。当任意

两个储罐中的化学物质液位过低时,每一个与门至少会有一个输入为低电平,从而使得它的输出也是低电平,这样反相器的最终输出 X 就是高电平。这个高电平输出随后被应用于驱动指示器。

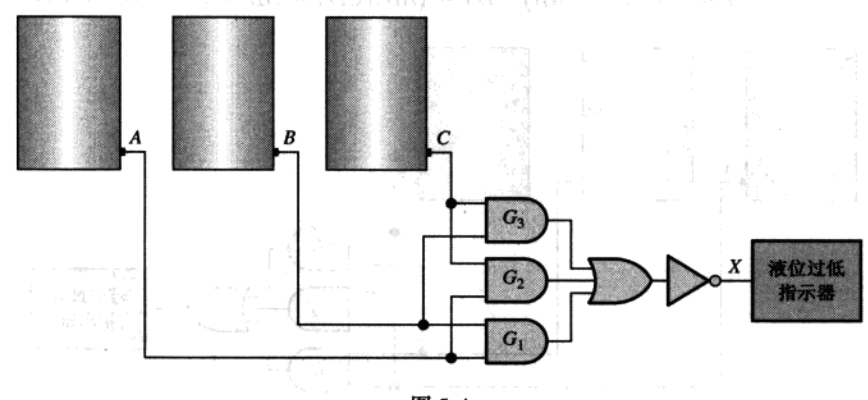


图 5.4

相关问题: 写出图 5.4 中与或非逻辑的布尔表达式,并指出当输入 A、B 和 C 中的任意两个都是低电平(0)时,输出就是高电平(1)。

5.1.3 异或逻辑

◇ 异或门实际上就是其他门的一个组合。

在第 3 章介绍了异或门。尽管由于它的重要性,这个电路被当做一种具有自己特殊符号的逻辑门,但是它实际上就是两个与门、一个或门和两个反相器的组合,如图 5.5(a)所示。两个 ANSI 标准逻辑符号如图 5.5(b)和图 5.5(c)所示。

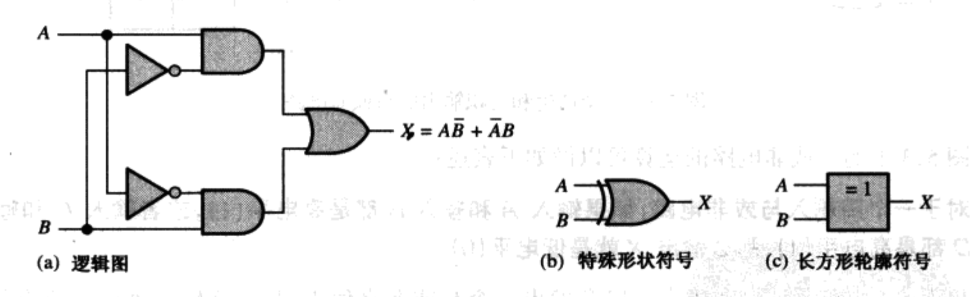


图 5.5 异或逻辑图和符号

图 5.5 中电路的输出表达式为

$$X = A\bar{B} + \bar{A}B$$

表 5.2 中的真值表是这个表达式的计算结果。注意只有在两个输入处于相反电平时,输出才是高电压。常常使用一个特殊异或运算符⊕,所以表达式 $X = A\bar{B} + \bar{A}B$ 可以表述为"X 等于 A 异或 B",并且可以写做

$$X = A \oplus B$$

表 5.2　异或逻辑的真值表

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

5.1.4　同或逻辑

正如所知道的那样，异或函数的反码就是同或函数，其推导过程如下所示：

$$X = \overline{A\overline{B} + \overline{A}B} = (\overline{A\overline{B}})(\overline{\overline{A}B}) = (\overline{A} + B)(A + \overline{B}) = \overline{A}\,\overline{B} + AB$$

注意只有当输入 A 和 B 处于相同的电平时，输出 X 才是高电平。

同或逻辑可以通过简单地将异或逻辑输出反相而实现，如图 5.6(a)所示，或者直接由表达式 $\overline{A}\overline{B} + AB$ 来实现，如图 5.6(b)所示。

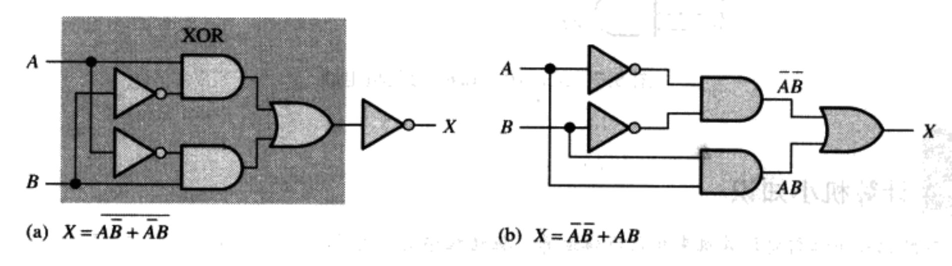


图 5.6　实现同或门的两种等价方法

5.2　组合逻辑电路的实现

在这一节中，将用一些示例来阐述怎样从布尔表达式或者真值表中实现逻辑电路。同时还讨论了使用第 4 章的方法把逻辑电路最小化。

学完本节以后，应当能够

- 从布尔表达式中实现逻辑电路
- 从真值表中实现逻辑电路
- 最小化一个逻辑电路

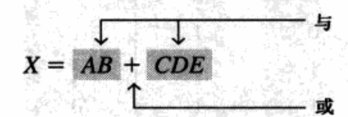
5.2.1　从布尔表达式到逻辑电路

◇ 每一个布尔表达式都有一个相应的逻辑电路，每一个逻辑电路也都有一个相应的布尔表达式。

看看下面这个布尔表达式：

$$X = AB + CDE$$

对该表达式进行简单的观察,就会发现这个表达式由两个项 AB 和 CDE 组成,并且有一个 5 变量的域。第一项由 A 和 B 相与组成,而第二项由 C、D 和 E 相与组成,这两项再进行或运算从而形成输出 X。这些运算由表达式的结构所指示,如下所示:



注意在这个特殊的表达式中,与运算形成的两个独立的项:AB 和 CDE,必须在这两项进行或运算之前执行。

为了实现这个布尔表达式,就需要一个二输入与门来形成项 AB,一个三输入与门来形成项 CDE。然后还需要一个二输入或门来组合这两个项。最后得到的逻辑电路如图 5.7 所示。

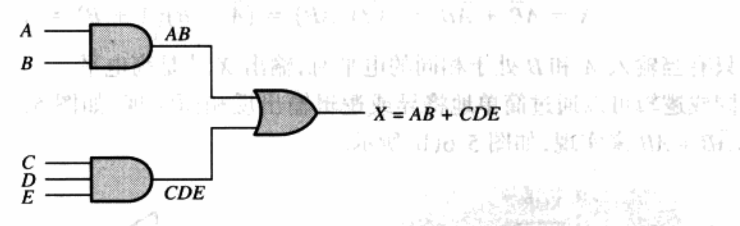


图 5.7　$X = AB + CDE$ 的逻辑电路



计算机小知识

许多控制程序都需要计算机来执行逻辑运算。驱动程序就是用于计算机外围设备的控制程序。例如,鼠标驱动程序需要逻辑检测来确定是否有按钮被按下,并且进一步的逻辑检测将确定其是否被移动,或者是水平方向或者是垂直方向。微处理器的核心部位是算术逻辑单元(ALU),其执行由程序指令所指定的逻辑运算。本章所描述的所有逻辑,如果有合适的指令,都可以由 ALU 来执行。

作为另一个例子,实现下面的表达式:

$$X = AB(C\overline{D} + EF)$$

该表达式的分解展示了项 AB 和 $C\overline{D} + EF$ 进行了与运算。项 $C\overline{D} + EF$ 通过对 C 和 $\overline{D}$ 及 E 和 F 首先进行与运算,然后再对这两项进行或运算。该结构和表达式的关系如下所示:

与
非
或

$$X = AB(C\overline{D} + EF)$$

与

在表达式最后形成之前,必须具有项 $C\overline{D} + EF$;但是在得到这个项之前,必须具有项 $C\overline{D}$ 和 EF;但是在得到项 $C\overline{D}$ 之前,必须具有项 $\overline{D}$。所以,正如所看到的那样,逻辑运算必须以恰当的次序来完成。

实现 $X = AB(C\overline{D} + EF)$所需要的逻辑门如下所示:

1. 一个反相器形成 $\overline{D}$；
2. 两个二输入与门形成 $C\overline{D}$ 和 EF；
3. 一个二输入或门形成 $C\overline{D}+EF$；
4. 一个三输入与门形成 X。

该表达式的逻辑电路如图 5.8(a)所示。注意在这个电路中(从输入 D 到输出)的输入和输出之间，最多有4个门和一个反相器。通过逻辑电路的总的传输延迟时间常常是一个主要的考虑因素。传输延迟是可添加的，所以输入和输出之间的门或者反相器越多，传输延迟时间就越大。

除了图 5.8(a)中的中间项 $C\overline{D}+EF$ 需要作为其他目的的输出之外，一般情况下最好将电路简化为它的乘积之和形式，以便减少总的传输延迟时间。该表达式被转换为如下所示的乘积之和形式，最后得到的电路如图 5.8(b)所示。

$$AB(C\overline{D}+EF)=ABC\overline{D}+ABEF$$

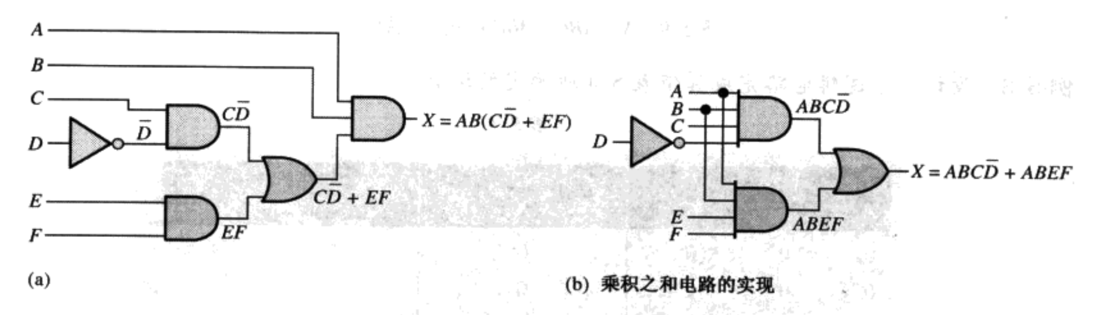


图 5.8 逻辑电路 $X=AB(C\overline{D}+EF)=ABC\overline{D}+ABEF$

5.2.2 从真值表到逻辑电路

如果开始于一个真值表而不是表达式，就可以从真值表中写出乘积之和表达式，然后实现逻辑电路。表 5.3 指定了一个逻辑函数。

表 5.3

输入			输出	
A	B	C	X	乘积项
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	0	

对于把乘积项进行或运算而从真值表中得到的布尔 SOP 表达式，当 $X=1$ 时有

$$X = \overline{A}BC + A\overline{B}\overline{C}$$

表达式中的第一项通过对三个变量 $\overline{A}$、B 和 C 相与而形成。第二项通过对三个量 A、$\overline{B}$ 和 $\overline{C}$ 相与而形成。

实现该表达式所需要的逻辑门如下所示：三个反相器以形成变量 $\overline{A}$、$\overline{B}$ 和 $\overline{C}$；两个三输入与门以形成项 $\overline{A}BC$ 和 $A\overline{B}\overline{C}$；以及一个二输入或门以形成最终的输出函数。该逻辑函数 $\overline{A}BC + A\overline{B}\overline{C}$ 的实现如图 5.9 所示。

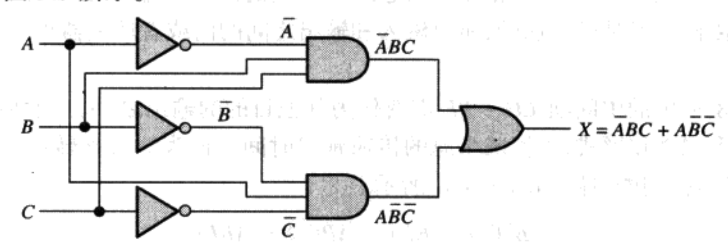


图 5.9　$X = \overline{A}BC + A\overline{B}\overline{C}$ 的逻辑电路

例 5.3　设计一个逻辑电路完成真值表 5.4 所确定的运算。

表 5.4

输入 A	B	C	输出 X	乘积项
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	0	
1	0	1	1	$A\overline{B}C$
1	1	0	1	$AB\overline{C}$
1	1	1	0	

解：注意仅在三个输入条件下 $X=1$。因此，逻辑表达式是

$$X = \overline{A}BC + A\overline{B}C + AB\overline{C}$$

逻辑门需要三个反相器，三个三输入与门和一个三输入或门。逻辑电路如图 5.10所示。

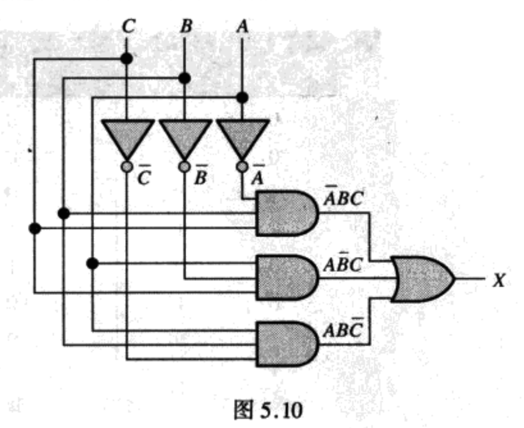


图 5.10

相关问题：判断图 5.10 中的逻辑电路是否能够被化简。

例 5.4　设计一个具有 4 个输入变量的逻辑电路，只有当 3 个输入变量全为 1 时，输出才为 1。

解:在4个变量的十六种组可能的组合中,其中有三个输入为1的组合列在了表5.5中,同时边上列出了相应的乘积项。

表5.5

A	B	C	D	乘积项
0	1	1	1	$\overline{A}BCD$
1	0	1	1	$A\overline{B}CD$
1	1	0	1	$AB\overline{C}D$
1	1	1	0	$ABC\overline{D}$

乘积项相或以后得到如下的表达式:

$$X = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D}$$

这个表达式由图5.11中的与或逻辑实现。

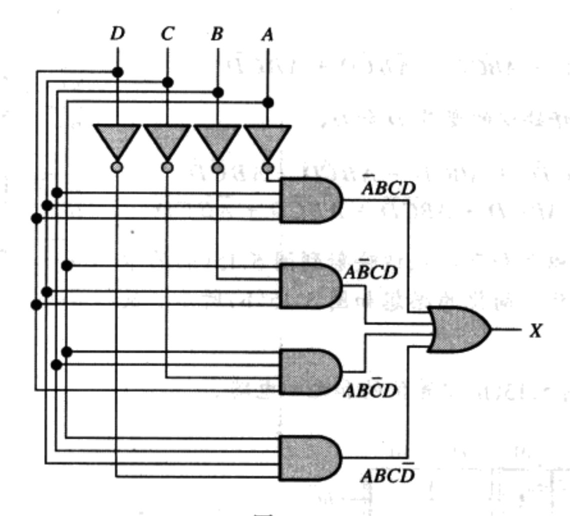


图5.11

相关问题:判断图5.11中的逻辑电路是否能够被化简。

例5.5 把图5.12中的组合逻辑电路简化到最小形式。

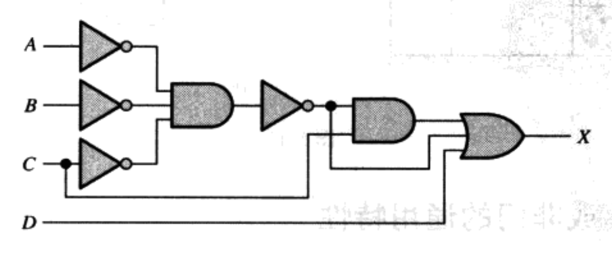


图5.12

解:输出电路的表达式是

$$X = (\overline{\overline{A}\,\overline{B}\,\overline{C}})C + \overline{\overline{A}\,\overline{B}\,\overline{C}} + D$$

应用狄摩根定理和布尔代数,

$$\begin{aligned} X &= (\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}})C + \overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}} + D \\ &= AC + BC + CC + A + B + C + D \\ &= AC + BC + C + A + B + C + D \\ &= C(A + B + 1) + A + B + D \\ X &= A + B + C + D \end{aligned}$$

如图 5.13 所示,简化的电路是一个四输入的或门。

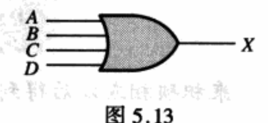


图 5.13

相关问题:使用卡诺图验证最小表达式 $A+B+C+D$。

例 5.6 最小化图 5.14 中的组合逻辑电路。求反变量的反相器没有画出。

解:输出表达式是

$$X = A\overline{B}\,\overline{C} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

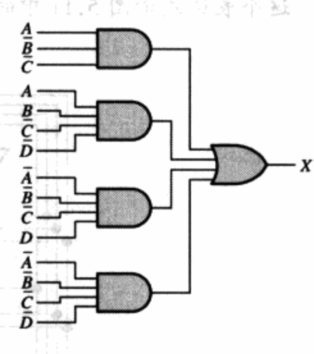


图 5.14

扩展第一项以包括缺少的变量 D 和 $\overline{D}$,

$$\begin{aligned} X &= A\overline{B}\,\overline{C}(D + \overline{D}) + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \\ &= A\overline{B}\,\overline{C}D + A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \end{aligned}$$

这个扩展了的乘积之和表达式被映射到图 5.15(a)的卡诺图上并得到简化。简化的思想如图 5.15(b)所示。反相器没有被画出。

相关问题:设计图 5.15(b)中等价的和之积电路。

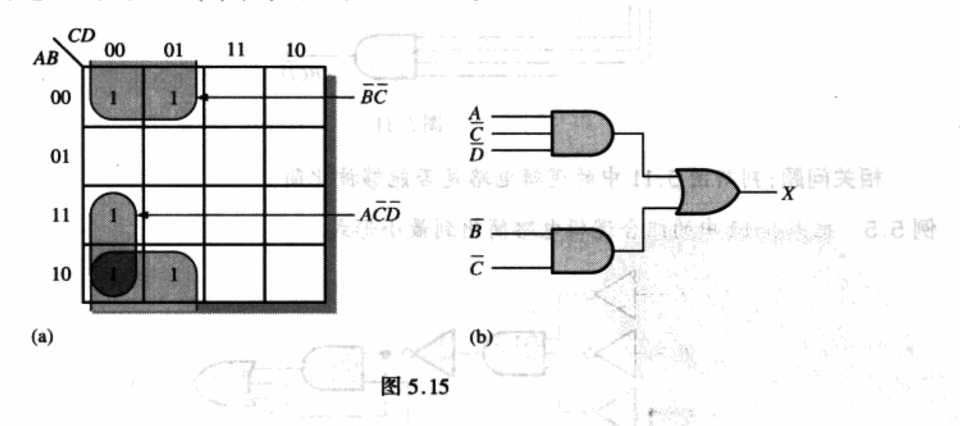


图 5.15

5.3 与非门和或非门的通用特性

到目前为止,我们已经学习了用与门、或门和反相器实现的组合电路。在这一节,我们讨论与非门(NAND)和或非门(NOR)的通用特性。与非门的通用特性是指它可以用做反相器,与

非门组合可以用来实现与、或和或非的运算。同样，或非门可以用来实现反相(非)、与、或和或非的运算。

学完本节以后，应该能够

- 使用与非门实现反相器、与门、或门和或非门
- 使用或非门实现反相器、与门、或门和或非门

5.3.1 与非门作为通用的逻辑元件

◇ 与非门可以用来产生任何逻辑函数。

与非门是一种通用门，因为它可以用来产生反相、与、或和或非的函数。反相器可以通过把与非门的所有输入连接在一起而形成，如图 5.16(a)所给出的二输入与非门，在效果上相当于一个输入。一个与函数可以仅由与非门产生，如图 5.16(b)所示。一个或函数可以仅由与非门产生，如图 5.16(c)所示。最后，一个或非函数的产生如图 5.16(d)所示。

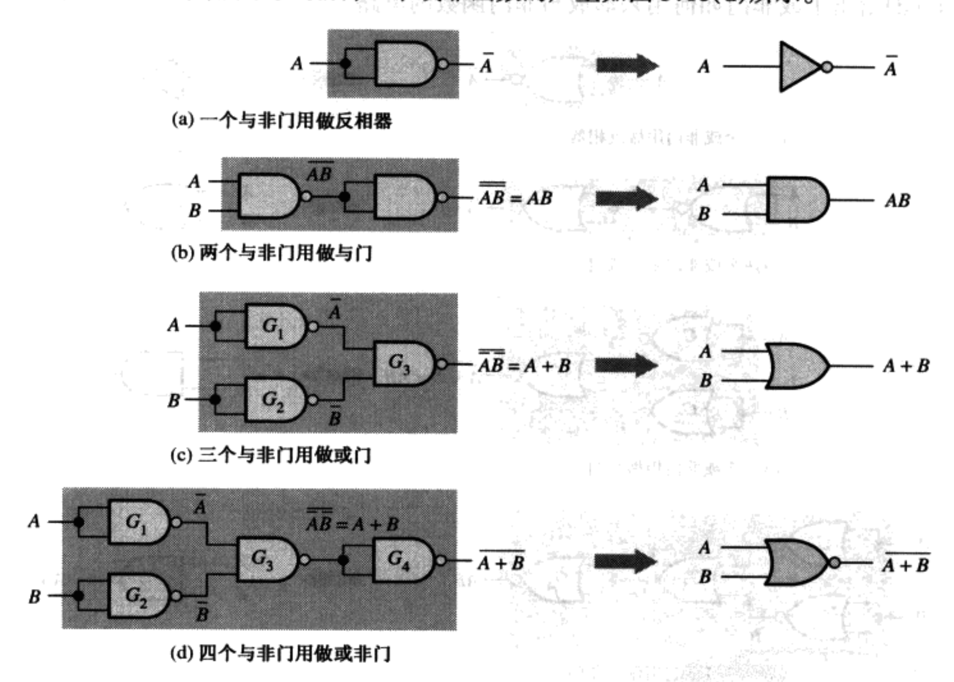


图 5.16 与非门的通用应用

在图 5.16(b)中，一个与非门用做把前面一个与非门的输出取反(反码)以形成与函数，如下面的等式所示：

$$X = \overline{\overline{AB}} = AB$$

在图 5.16(c)中，门 G_1 和门 G_2 用做在两个输入变量进入与非门 G_3 以前取反。最后的或输出应用狄摩根定理推导如下：

$$X = \overline{\overline{A}\,\overline{B}} = A + B$$

在图 5.16(d)中,与非门 G_4 用做反相器,连接到图 5.16(c)的电路上以产生或非运算 $\overline{A+B}$。

5.3.2 或非门作为通用的逻辑元件

◇ 或非门可以用来产生任何逻辑函数。

如同与非门,或非门它可以用来产生反相、与、或和与非的函数。一个非电路,或反相器可以通过把或非门的所有输入连接在一起而形成,如图 5.17(a)所给出的二输入或非门,在效果上相当于一个输入。同样,一个或门可以由或非门产生,如图 5.17(b)所示。一个与门可以通过使用或非门构造出来,如图 5.17(c)所示。这里或非门 G_1 和 G_2 用做反相器,最后的与输出应用狄摩根定理推导如下:

$$X = \overline{\overline{A} + \overline{B}} = AB$$

图 5.17(d)给出了或非门如何用来形成与非门函数的电路。

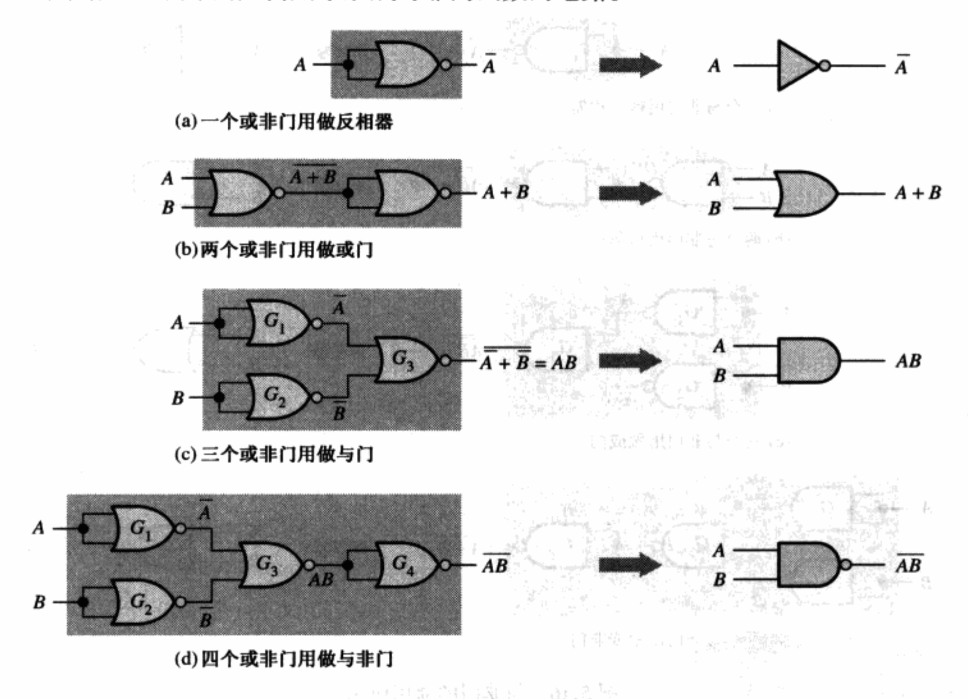


图 5.17 或非门的通用应用

5.4 使用与非门和或非门的组合逻辑

在这一节中,将会看到与非门和或非门怎样用来实现一个逻辑函数。回顾第 3 章,与非门也可以呈现一个称为非 - 或的等价运算,而或非门可以呈现一个称为非 - 与的等价运算。下面将会看到怎样使用合适的符号来表示等价运算,以使逻辑图更加容易"读"。

学完本节以后,应当能够

- 使用与非门来实现逻辑函数
- 使用或非门来实现逻辑函数
- 在逻辑图中使用合适的双符号

5.4.1 与非逻辑

正如我们已经学过的，通过使用狄摩根定理，与非门可以作为与非或者非-或函数。

$$\overline{AB} = \overline{A} + \overline{B}$$

与非 ↗ ↖ 非-或

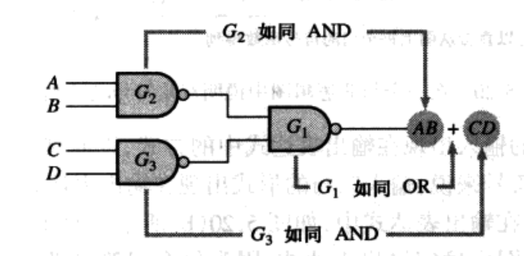
考虑图5.18中的与非逻辑。设计输出表达式的步骤如下所示：

$$\begin{aligned} X &= \overline{(\overline{AB})(\overline{CD})} \\ &= \overline{(\overline{A}+\overline{B})(\overline{C}+\overline{D})} \\ &= (\overline{\overline{A}+\overline{B}}) + (\overline{\overline{C}+\overline{D}}) \\ &= \overline{\overline{A}}\,\overline{\overline{B}} + \overline{\overline{C}}\,\overline{\overline{D}} \\ &= AB + CD \end{aligned}$$

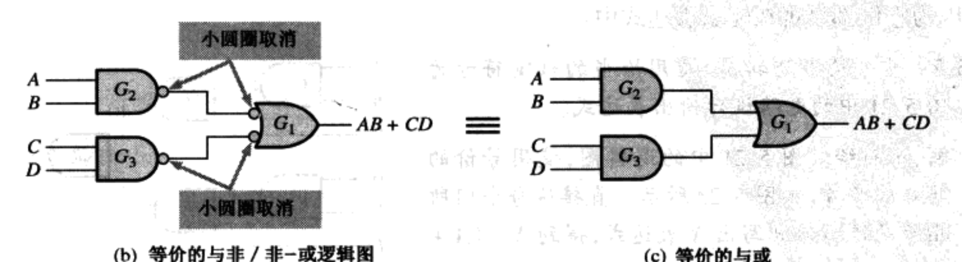
图5.18 $X = AB + CD$ 的与非逻辑

正如在图5.18中所看到的一样，输出表达式 $AB + CD$ 是两个与项被或在了一起。图5.19(a)给出了门 G_2 和 G_3 用做与门、门 G_1 用做或门的情况。这个电路在图5.19(b)中重新绘制了出来，对 G_2 和 G_3 使用了与非符号，而对门 G_1 使用了非-或符号。

注意在图5.19(b)中门 G_2 和 G_3 的输出和门 G_1 的输入之间的小圆圈到小圆圈的连接：由于一个小圆圈表示一个反相，两个连接的小圆圈就表示一个双重反相，所以相互抵消。这种反相抵消可以在前面输出表达式 $AB + CD$ 的设计中看到，并且由输出表达式中上划线的省略所指示，如图5.19(c)所示。



(a) 原始的与非逻辑图给出了和输出表达式有关的有效门运算

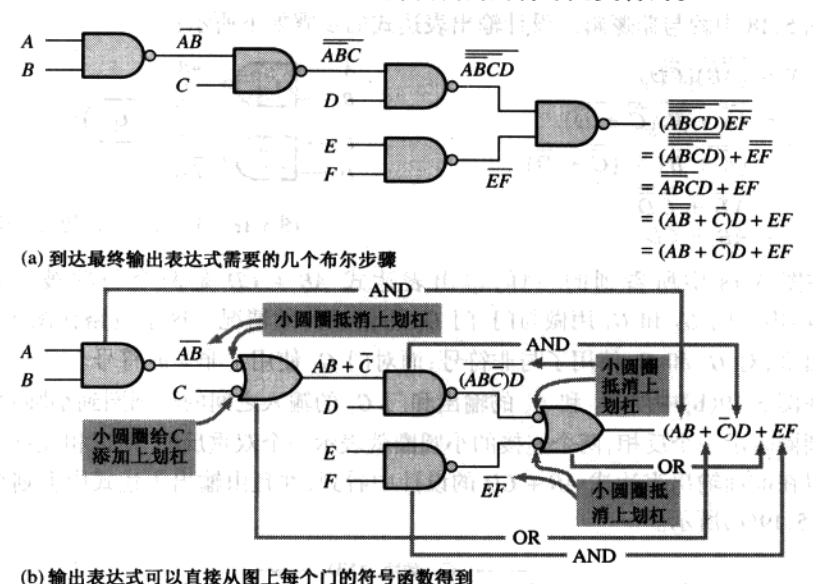


(b) 等价的与非/非-或逻辑图　　(c) 等价的与或

图5.19 图5.18中电路的与或等价逻辑的设计

使用双符号的与非逻辑图 对于所有使用与非门的逻辑图,在绘制它们的时候,都要用一个与非门符号或者等价的非-或符号来表示每一个门,以反映逻辑电路中间门的运算。与非符号和非-或符号称为双重符号。在绘制一个与非逻辑图时,总是以这样的方法使用门的符号,在一个门的输出和另一个门的输入之间的每个连接要么是小圆圈到小圆圈,要么是非小圆圈到非小圆圈。在逻辑图中,一个小圆圈输出不应当连接到非小圆圈输入,反之亦然。

图 5.20 给出了门的一种排列,用以说明为具有几个门级的与非电路使用合适的双重符号的过程。尽管像图 5.20(a)那样使用所有的与非符号是正确的,但图 5.20(b)中的图就更加容易“读”,因而是更好的方法。如图 5.20(b)所示,输出门以非-与符号出现。这时,与非符号恰好用在输出门的前面,并且当从输出移开时,接续门的符号是交替的。



(a) 到达最终输出表达式需要的几个布尔步骤

(b) 输出表达式可以直接从图上每个门的符号函数得到

图 5.20 在一个与非逻辑图中说明双重符号的恰当使用

门的形状指示了它的输入出现在输出表达式中的方式,因此展示了该门的功能在逻辑电路中的作用。对于与非符号来说,输入以与的形式出现在输出表达式中;而对于非-或符号来说,输入以或的形式出现在输出表达式中,如图 5.20(b)所示。对于图 5.20(b)中的双重符号图,很容易直接从该逻辑图中确定输出表达式,因为每个门符号都指示了它的输入变量的关系,因为它们都呈现在输出表达式中。

例 5.7 重新绘制逻辑图,使用恰当的双重符号为图 5.21 中的电路设计输出表达式。

解: 重新绘制图 5.21 中的逻辑图,使用等价的非-或符号,如图 5.22 所示。直接从每个门所指示的逻辑运算写出 X 表达式,得到 $X=(\overline{A}+\overline{B})C+(\overline{D}+\overline{E})F$。

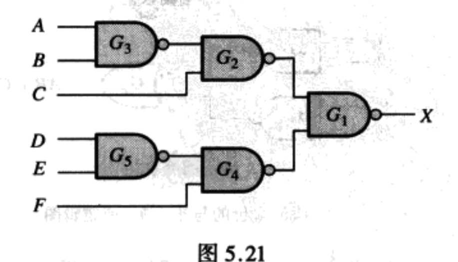


图 5.21

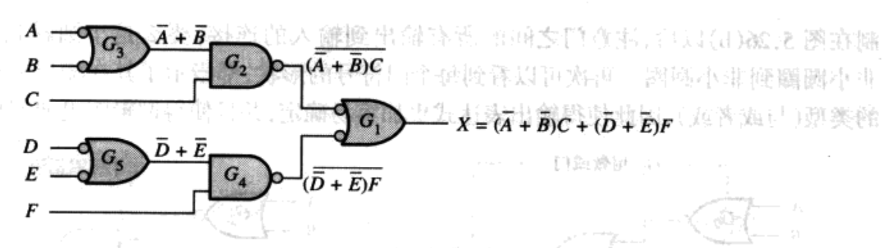


图 5.22

相关问题：从图 5.21 中导出输出表达式，并展示它等价于解中的表达式。

例 5.8 使用恰当的双重符号实现每一个与非逻辑表达式：

(a) $ABC + DE$　　　　(b) $ABC + \overline{D} + \overline{E}$

解：参照图 5.23。

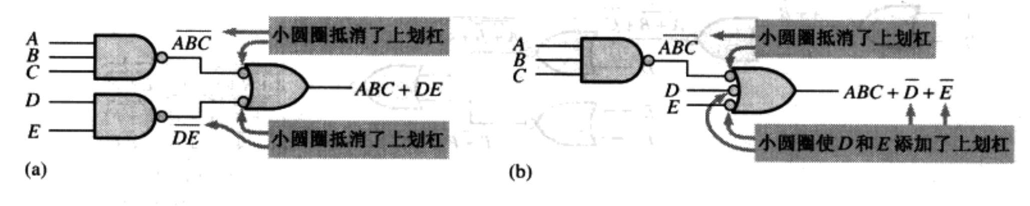


图 5.23

相关问题：把图 5.23(a)和图 5.23(b)中的与非电路转换为等价的非－或逻辑。

5.4.2 或非逻辑

或非门可以用做或非或者非－与，如同狄摩根定理所展示的那样。

$$\overline{A+B} = \overline{A}\,\overline{B}$$

或非　　　　非-与

考虑一下图 5.24 中的或非逻辑。输出表达式的设计如下所示：

$$X = \overline{\overline{A+B}+\overline{C+D}} = (\overline{\overline{A+B}})(\overline{\overline{C+D}}) = (A+B)(C+D)$$

图 5.24　或非逻辑 $X=(A+B)(C+D)$

如在图 5.24 所见到的，输出表达式$(A+B)(C+D)$由两个或项的相与组成。图 5.25(a)展示了门 G_2 和 G_3 用做或门，而门 G_1 用做与门。这个电路在图 5.25(b)中用门 G_1 的非－与符号重新绘制。

使用双重符号的与非逻辑图　和与非逻辑相同，使用双重符号的目的是使得逻辑图更加容易读和分析，如图 5.27 中或非逻辑电路所示。当图 5.26(a)中的电路利用双重符号重新绘

制在图 5.26(b)以后，注意门之间的所有输出到输入的连接，要么是小圆圈到小圆圈，要么是非小圆圈到非小圆圈。再次可以看到每个门符号的形状，都指示了其在输出表达式中所产生项的类型(与或者或)，因此使得输出表达式更加容易确定，并且使得逻辑图更加容易分析。

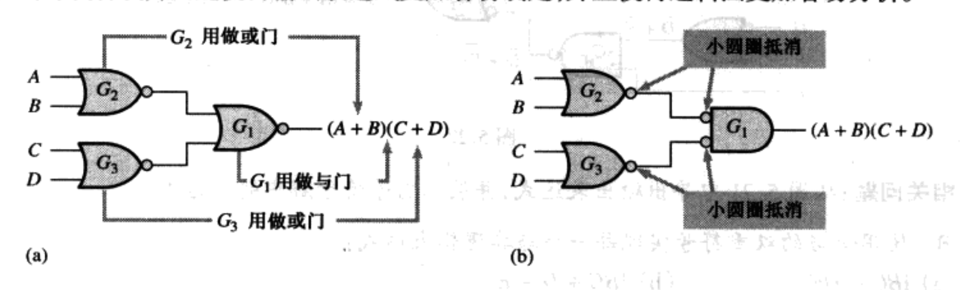
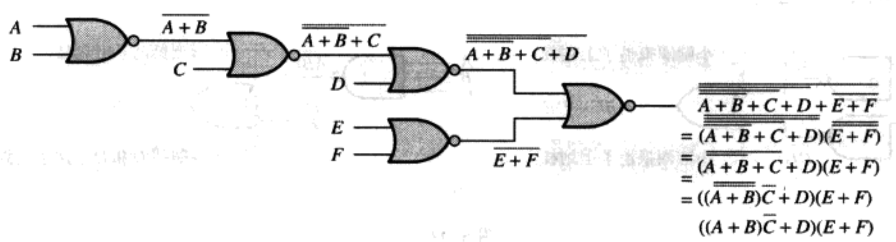
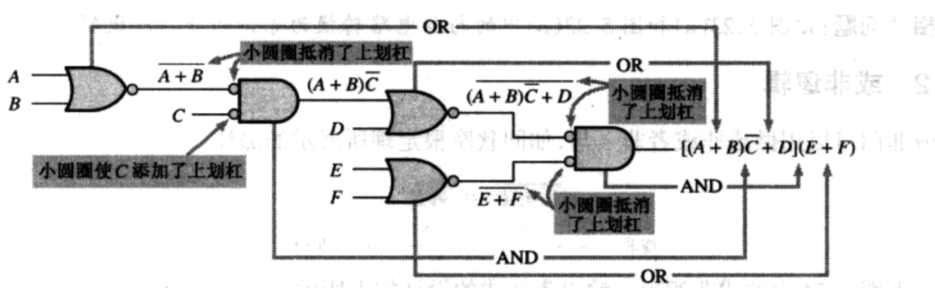


图 5.25



(a) 在经过几个布尔步骤之后，得到了最终的输出表达式



(b) 输出表达式可以直接从图中每个门符号的函数中得到

图 5.26 在或非逻辑图中使用恰当的双重符号

例 5.9 使用恰当的双重符号，重新绘制逻辑图并设计图 5.27 中电路的输出表达式。

解: 使用等价的非－与符号重新绘制逻辑图，如图 5.28 所示，直接从每个门所指示的运算中写出 X 的表达式。

$$X = (\overline{A}\,\overline{B} + C)(\overline{D}\,\overline{E} + F)$$

图 5.27

相关问题: 证明图 5.27 中或非电路的输出和图 5.28中电路的输出是一样的。

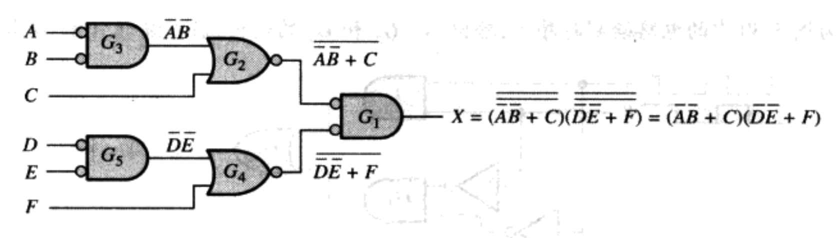


图 5.28

5.5 具有脉冲波形输入的逻辑电路运算

本节介绍了几个具有脉冲波形输入的一般性组合逻辑电路的例子。要记住每个门的运算对于脉冲输入和常量电平输入都是相同的。在任何给定时间的逻辑电路输出取决于那个特定时刻的输入,所以输入随时间变化的关系就显得尤为重要。

学完本节以后,应当能够

- 分析具有脉冲波形输入的组合逻辑电路
- 为具有特定输入的任意给定组合逻辑电路绘制时序图

任何门的运算都是相同的,不管它的输入是脉冲还是常量电位。输入的本质(脉冲或者常量电位)并不会改变电路的真值表。本节中的例子将阐述具有脉冲输入的组合逻辑电路的分析。

下面是独立门运算的回顾,用以分析具有脉冲波形输入的组合电路:

1. 只有当所有的输入在相同的时刻都是高电平时,与门的输出才是高电平;
2. 只要有一个输入是高电平时,或门的输出就是高电平;
3. 只有所有的输入在相同的时刻都是高电平时,与非门的输出才是低电平;
4. 只要有一个输入是高电平时,或非门的输出就是低电平。

例 5.10 确定图 5.29 中电路的最终输出波形 X,输入波形 A、B 和 C 如图所示。

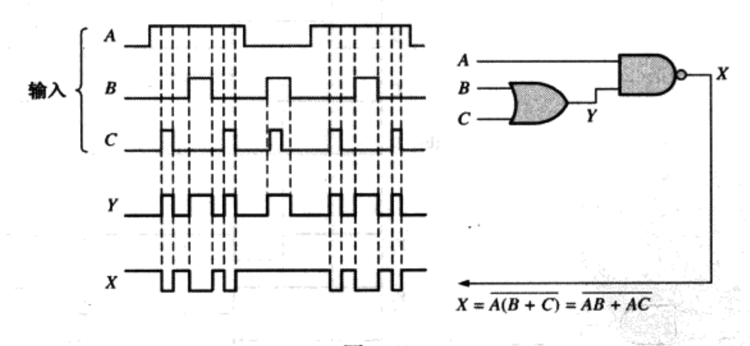


图 5.29

解: 输出表达式 $\overline{AB+AC}$ 指示了当 A 和 B 都是高电平或者当 A 和 C 都是高电平或者当所有的输入都是高电平时,输出 X 就是低电平。该输出波形 X 如图 5.29 中的时序图所示。同时还给出了或门输出处中间的波形 Y。

相关问题: 如果输入 A 是一个常量高电平,那么确定其输出波形。

例 5.11 为图 5.30 中的电路绘制时序图，给出 G_1、G_2 和 G_3 的输出，输入波形 A 和 B 如图所示。

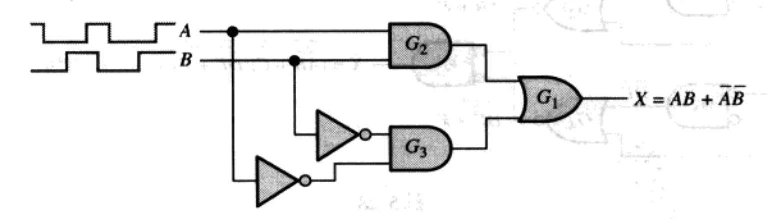


图 5.30

解：当两个输入都是高电平或者当两个输入都是低电平时，输出 X 就是高电平，如图 5.31 所示。注意这是一个同或门电路。门 G_2 和 G_3 的中间输出也在图 5.31 中给出。

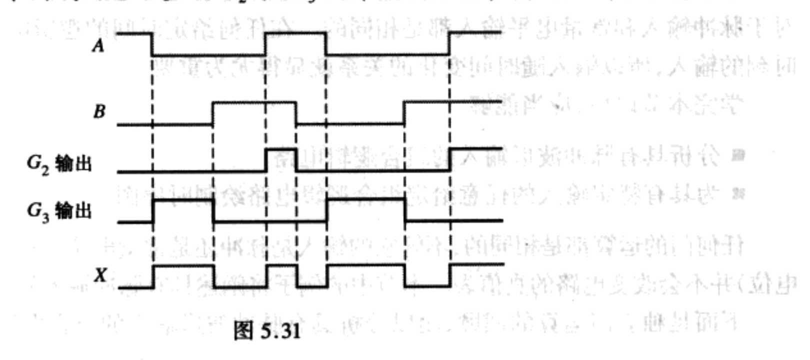


图 5.31

相关问题：如果输入 B 被反相，请确定图 5.30 中的输出 X。

例 5.12 为图 5.32(a)中所示的逻辑电路确定输出波形 X，首先找到位于每一个点 Y_1、Y_2、Y_3 和 Y_4 上的中间波形。输入波形如图 5.32(b)所示。

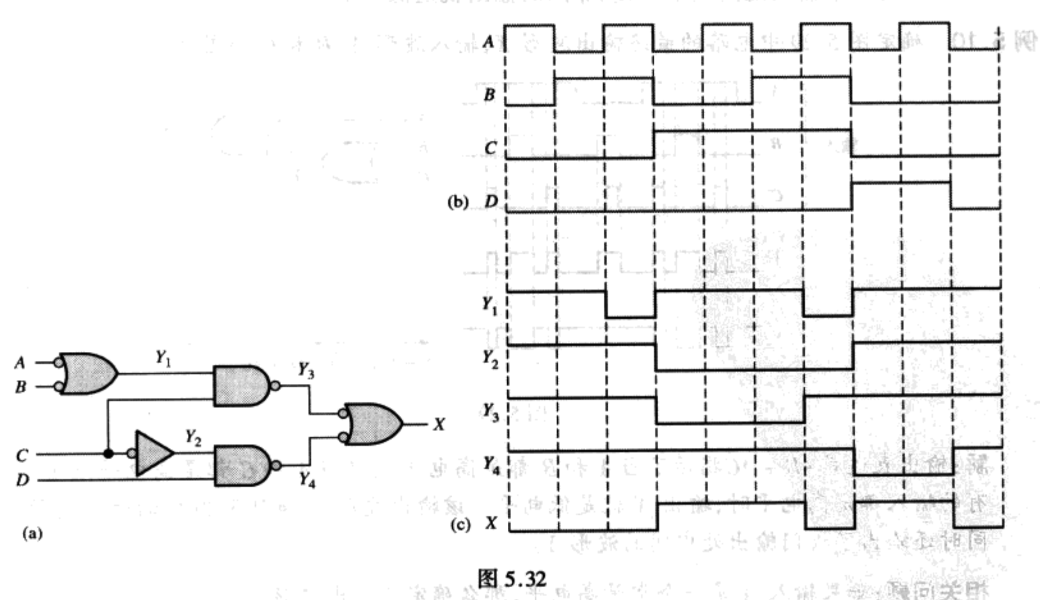


图 5.32

解：所有的中间波形及最终输出波形都在图5.32(c)中给出。

相关问题：如图输入波形 A 被反相，请确定 Y_1、Y_2、Y_3 和 Y_4 的波形。

例 5.13 为例5.12中图5.32(a)中的电路确定输出波形 X，直接从输出表达式中确定。

解：该电路的输出表达式的设计过程如图5.33所示。这个乘积之和形式指示了当 A 为低电平及 C 为高电平，或者 B 为低电平及 C 为高电平，或者 C 为低电平而 D 为高电平时，输出就是高电平。

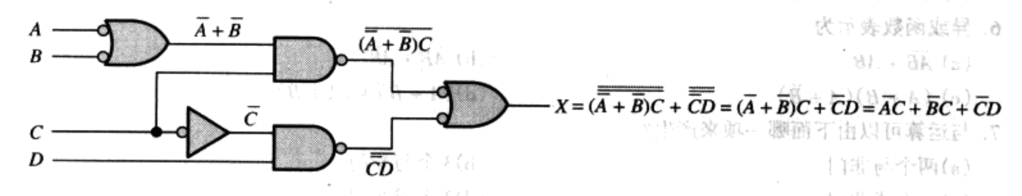


图 5.33

结果如图5.34所示，并且和例5.12中用中间波形方法得到的结果一样。每个产生高电平输出的波形条件所对应的乘积项都展示了出来。

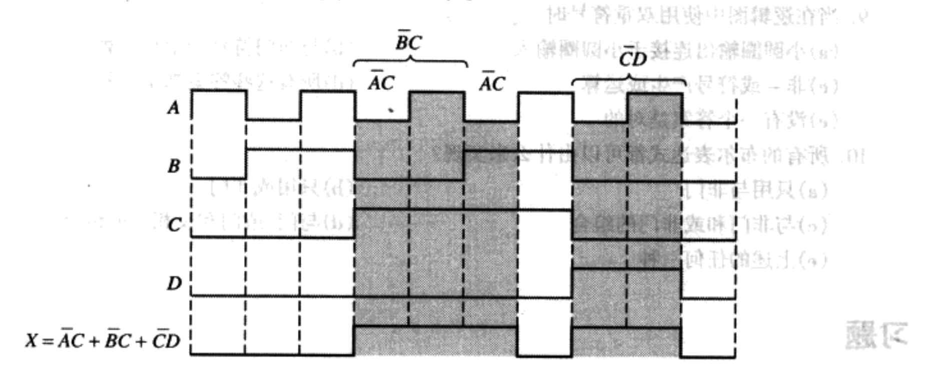


图 5.34

相关问题：如果输入波形全部反相，重复此例。

自测题 (答案在本章的结尾)

1. 一个与或电路，具有一个输入为 A、B、C 和 D 的与门和一个输入为 E 和 F 的与门，那么该电路的输出表达式为
   (a) $ABCDEF$　　(b) $A+B+C+D+E+F$
   (c) $(A+B+C+D)(E+F)$　　(d) $ABCD+EF$
2. 输出 $X=A\overline{B}C+A\overline{C}$ 的逻辑电路由________组成。
   (a) 两个与门和一个或门　　(b) 两个与门、一个或门和两个反相器
   (c) 两个或门、一个与门和两个反相器　　(d) 两个与门、一个或门和一个反相器
3. 为了实现表达式 $\overline{A}BCD+A\overline{B}CD+AB\overline{C}\overline{D}$，需要一个或门和________。
   (a) 一个与门　　(b) 3个与门

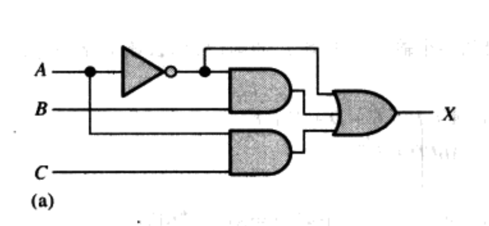
(c) 3 个与门和 4 个反相器
(d) 3 个与门和 3 个反相器

4. 表达式 $\bar{A}BCD + ABC\bar{D} + A\bar{B}\bar{C}D$
   (a)不能简化
   (b)可以简化为 $\bar{A}BC + A\bar{B}$
   (c)可以简化为 $ABC\bar{D} + \bar{A}B\bar{C}$
   (d)这些答案都不对
5. 某个与或非电路具有一个输入为 A、B、C 和 D 的与门,以及一个输入为 E 和 F 的与门,那么它的输出表达式为
   (a) $ABCD + EF$
   (b) $\bar{A} + \bar{B} + \bar{C} + \bar{D} + \bar{E} + \bar{F}$
   (c) $\overline{(A + B + C + D)(E + F)}$
   (d) $(\bar{A} + \bar{B} + \bar{C} + \bar{D})(\bar{E} + \bar{F})$
6. 异或函数表示为
   (a) $\bar{A}\bar{B} + AB$
   (b) $\bar{A}B + A\bar{B}$
   (c) $(\bar{A} + B)(A + \bar{B})$
   (d) $(\bar{A} + \bar{B}) + (A + B)$
7. 与运算可以由下面哪一项来产生?
   (a)两个与非门
   (b)3 个与非门
   (c)一个或非门
   (d)3 个或非门
8. 或运算可以由下面哪一项来产生?
   (a)两个或非门
   (b)3 个与非门
   (c)4 个与非门
   (d)答案(a)和(b)
9. 当在逻辑图中使用双重符号时________。
   (a)小圆圈输出连接于小圆圈输入
   (b)与非门符号产生与运算
   (c)非 - 或符号产生或运算
   (d)所有这些答案都是对的
   (e)没有一个答案是对的
10. 所有的布尔表达式都可以由什么来实现?
   (a)只用与非门
   (b)只用或非门
   (c)与非门和或非门的组合
   (d)与门、或门和反相器的组合
   (e)上述的任何一种

习题

5.1 节　基本组合逻辑电路

1. 为 3-wide(每个设备具有三个与门)四输入的与或非电路绘制 ANSI 特殊形状逻辑图。同时绘制 ANSI 标准矩形轮廓符号。
2. 为图 5.35 中的每一个电路写出输出表达式。



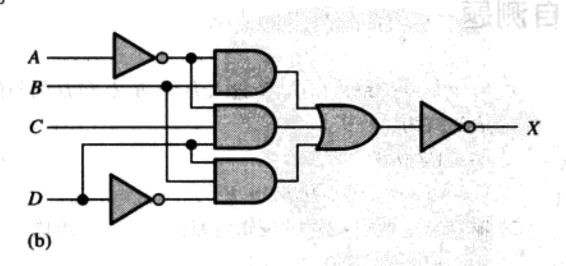


图 5.35

3. 为图 5.36 中的每一个电路写出输出表达式。

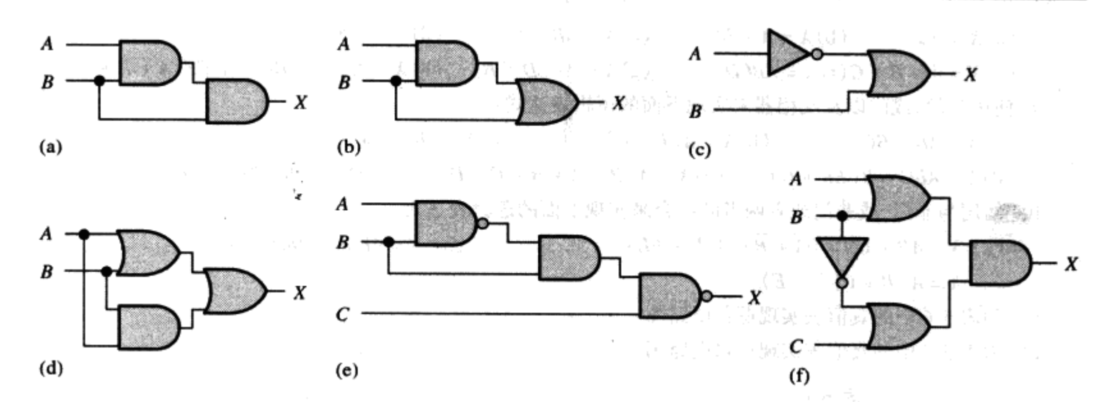


图 5.36

4. 为图 5.37 中的每一个电路写出输出表达式,然后将每个电路改变为等价的与或形式。

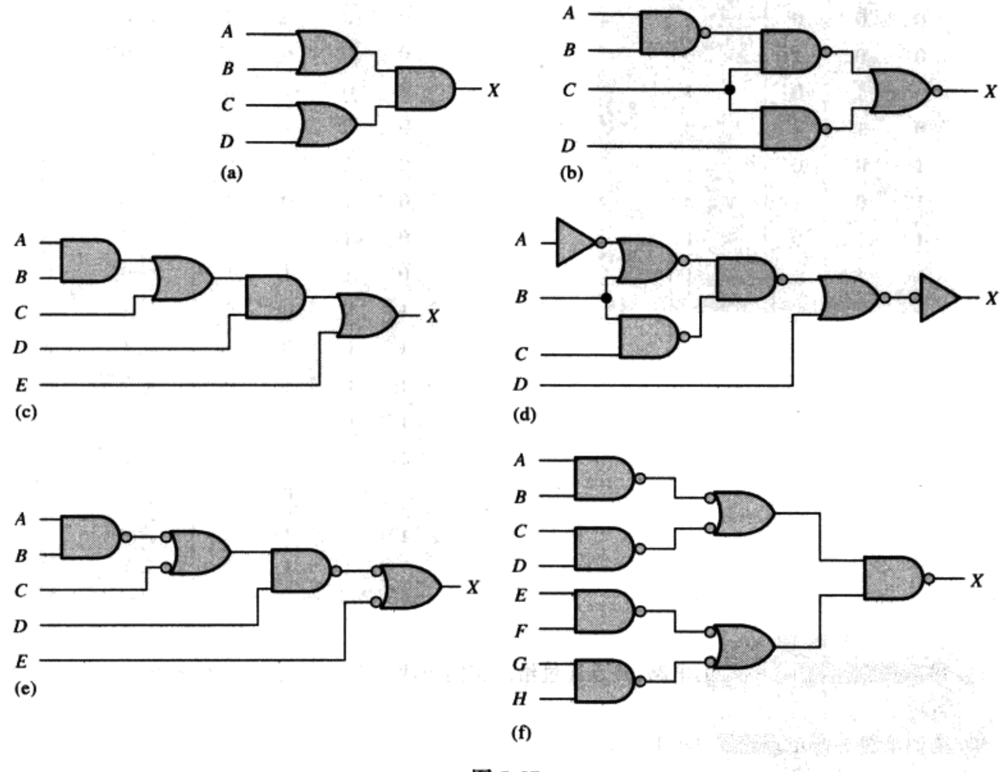


图 5.37

5. 为图 5.36 中的每一个电路开发出真值表。

6. 为图 5.37 中的每一个电路开发出真值表。

7. 给出一个产生和之积输出的同或门电路。

5.2 节 组合逻辑电路的实现

8. 使用与门、或门或者两者的组合实现下面的逻辑表达式:

(a) $X = AB$ (b) $X = A + B$ (c) $X = AB + C$ (d) $X = ABC + D$

(e) $X = A + B + C$ (f) $X = ABCD$ (g) $X = A(CD + B)$ (h) $X = AB(C + DEF) + CE(A + B + F)$

9. 使用与门、或门以及反相器来实现下面的逻辑表达式：

   (a) $X = AB + \bar{B}C$ (b) $X = A(B + \bar{C})$ (c) $X = A\bar{B} + AB$

   (d) $X = \overline{ABC} + B(EF + \bar{G})$ (e) $X = A[BC(A + B + C + D)]$ (f) $X = B(C\bar{D}E + \bar{E}FG)(\overline{AB} + C)$

10. 使用与非门、或非门或者两者的组合来实现下面的逻辑表达式：

    (a) $X = \bar{A}B + CD + (\overline{A + B})(ACD + \overline{BE})$ (b) $X = AB\bar{C}\bar{D} + D\bar{E}F + \overline{AF}$

    (c) $X = \bar{A}[B + \bar{C}(D + E)]$

11. 为表 5.6 中的真值表实现逻辑电路图。

12. 为表 5.7 中的真值表实现逻辑电路图。

表 5.6

输入			输出
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

表 5.7

输入				输出
A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

13. 尽可能地化简图 5.38 中的电路，并且通过给出相同的两个真值表，验证简化的电路和原来的电路等价。

14. 为图 5.39 中的电路重复习题 13。

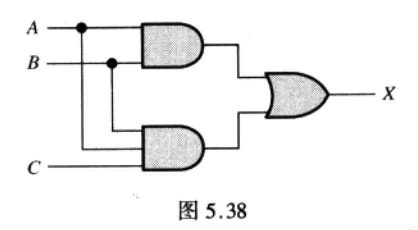


图 5.38

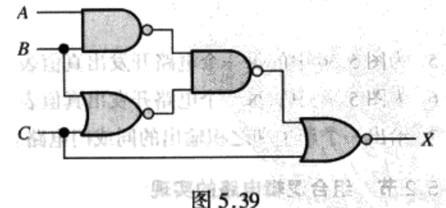


图 5.39

15. 以乘积项之和的形式,使用最少的门实现习题9中的每个函数。
16. 以乘积项之和的形式,使用最少的函数实现习题10中的每个函数。
17. 以乘积项之和的形式,使用最少的函数实现图5.37中每个电路的函数。

5.3节 与非门和或非门的通用特性

18. 只用与非门来实现图5.35中的逻辑电路。
19. 只用与非门来实现图5.39中的逻辑电路。
20. 只使用或非门重复习题18。
21. 只使用或非门重复习题19。

5.4节 使用与非门和或非门的组合逻辑

22. 给出怎样只使用或非门实现下面的表达式的方法:
    (a) $X = ABC$ (b) $X = \overline{ABC}$ (c) $X = A + B$
    (d) $X = A + B + \overline{C}$ (e) $X = \overline{AB} + \overline{CD}$ (f) $X = (A + B)(C + D)$
    (g) $X = AB[C(\overline{DE} + \overline{AB}) + \overline{BCE}]$
23. 只使用与非门重复习题22。
24. 只使用与非门来实现习题8中的每一个函数。
25. 只使用与非门来实现习题9中的每一个函数。

5.5节 具有脉冲波形输入的逻辑电路运算

26. 给定如图5.40所示的逻辑电路和输入波形,绘制输出波形。
27. 对于图5.41中的逻辑电路,绘制出正确的与输入相关的输出波形。

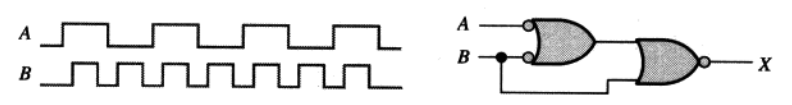


图5.40

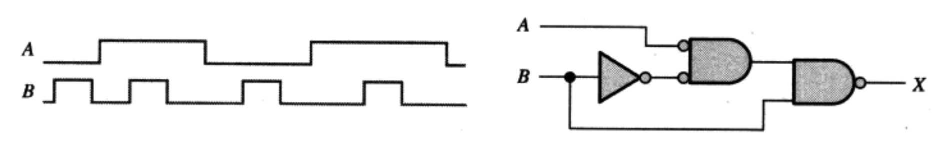


图5.41

28. 对于图5.42中的输入波形,什么逻辑电路将会产生所给出的输出波形?
29. 为图5.43中的波形重复习题28。

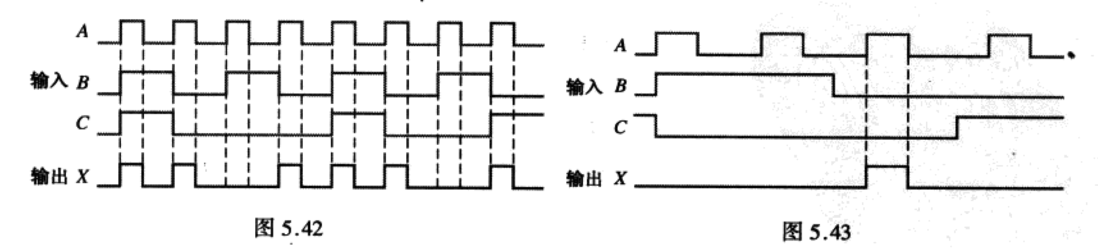


图5.42 图5.43

30. 对于图5.44中的电路,按照适当的相互关系绘制标有数字地方的波形。
31. 假设穿过每个门的传输延迟是10纳秒(ns),确定图5.45中(具有最小宽度 $t_W = 25$ ns 的脉冲如图标出)所需要的输出波形 X 是否可以由给出的输入来产生。

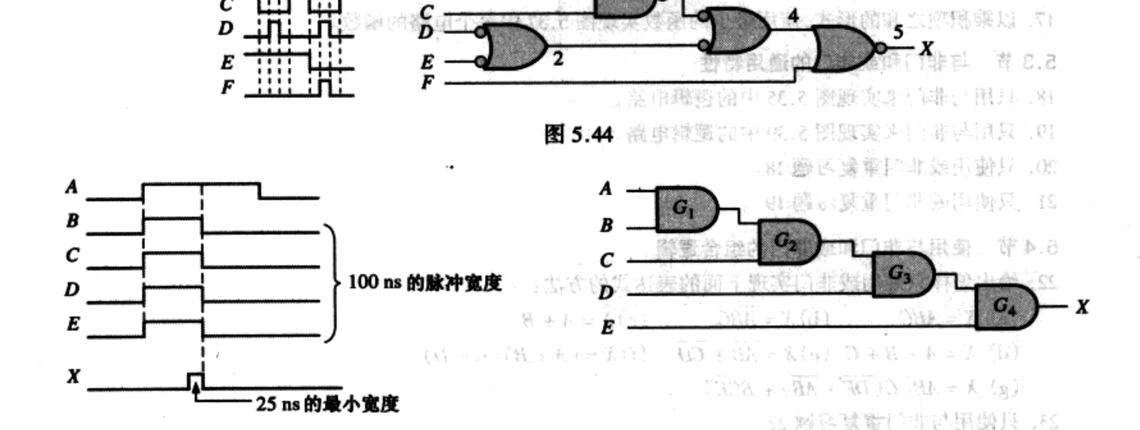


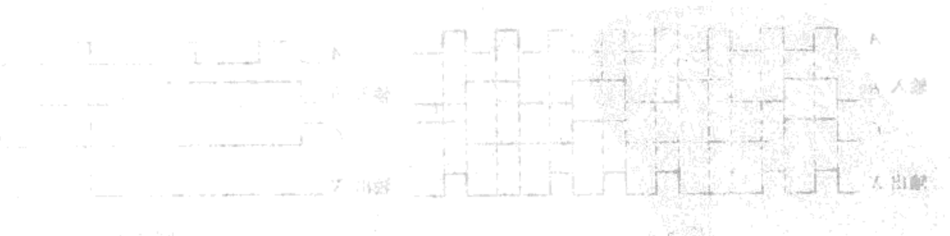
图 5.44



图 5.45

自测题答案

1.(d) 2.(b) 3.(c) 4.(a) 5.(d) 6.(b) 7.(a) 8.(d) 9.(d) 10.(e)



第6章　组合逻辑电路函数

章节提纲

6.1　基本加法器

加法器在计算机中很重要,并且在其他类型的处理数字数据的数字系统中也很重要。如果要学习数字系统,那么理解基本加法器的运行是学习的基础。在本节的内容里,我们将介绍半加器和全加器。

学完本节以后,应当能够

- 描述半加器的函数
- 绘制半加器的逻辑图
- 描述全加器的函数
- 利用半加器绘制全加器的逻辑图
- 使用与或逻辑实现一个全加器

6.1.1　半加器

◇ *一个半加器进行两位相加,产生一个和及一个进位输出。*

回顾第2章中所讲到的二进制相加的基本规则:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

实现半加运算的逻辑电路称为半加器(half-adder)。

一个半加器的输入为两个二进制数,并在输出端产生两个二进制数、一个和位和一个进位位。

如图 6.1 所示是用逻辑符号表示的一个半加器。

半加器逻辑 根据表 6.1 所给出的半加器的运算,和及进位作为输入函数的表达式可以通过推导获得。注意只有当 A 和 B 都为 1 的时候,进位输出 C_{out} 才为 1;所以,C_{out} 可表示为输入变量的与:

$$C_{out} = AB \tag{6.1}$$

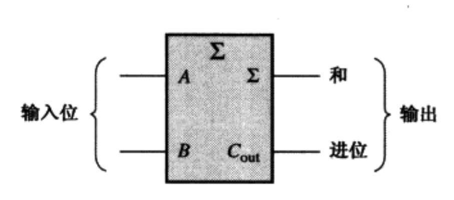
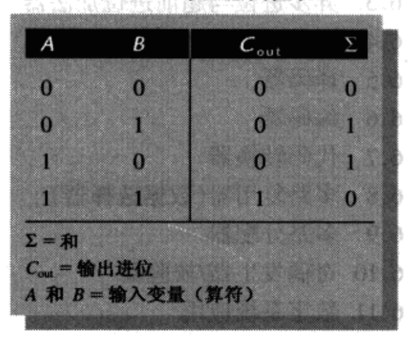


图 6.1 半加器的逻辑符号

表 6.1 半加器真值表



A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = 和
C_{out} = 输出进位
A 和 B = 输入变量(算符)

现在请注意,只有当输入变量 A 与 B 不相等的时候,和输出(Σ)才为 1。所以和可以用输入变量的异或来表示:

$$\Sigma = A \oplus B \tag{6.2}$$

根据式(6.1)和式(6.2),可以设计出满足半加器功能的逻辑电路。A 和 B 输入与门产生进位输出,异或门产生一个和输出,如图 6.2 所示。记住,异或门是由与门、或门和反相器实现的。

6.1.2 全加器

◇ 全加器有一个进位输入,而半加器没有。

另一类加法器就是全加器(full-adder)。

全加器有两个输入位和一个进位输入,它产生一个和输出及一个进位输出。

全加器和半加器的主要区别就是全加器有一个进位输入。全加器的逻辑符号如图 6.3 所示,全加器运算的真值表如表 6.2 所示。

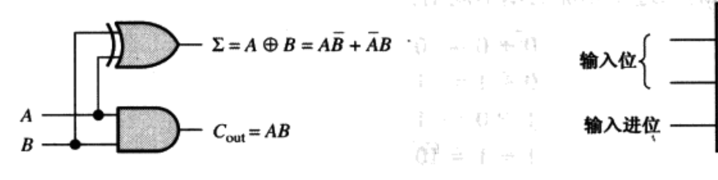


图 6.2 半加器逻辑图

图 6.3 全加器的逻辑符号

全加器逻辑　全加器必须将两个输入位和一个进位输入位相加。从半加器中可以知道，输入位 A 和 B 的和就是这两个变量的异或 $A \oplus B$。由于进位输入（C_{in}）是与这两个输入位相加，那么它一定是和 $A \oplus B$ 异或，这样就得到全加器输出和的等式：

$$\Sigma = (A \oplus B) \oplus C_{in} \tag{6.3}$$

这就意味着要实现全加器的和函数，需要使用两个二输入异或门，第一个产生项 $A \oplus B$，第二个使用第一个异或门的输出和进位输入作为输入，如图 6.4(a)所示。

表 6.2　全加器真值表

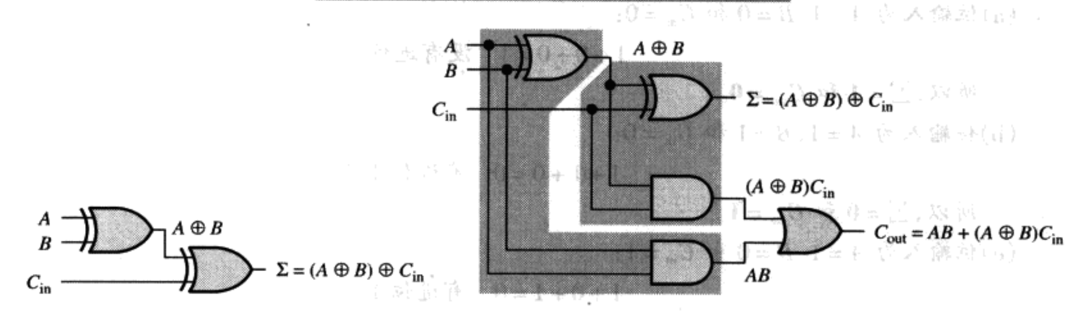
A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = 进位输入，有时定义为 CI

C_{out} = 输出进位，有时定义为 CO

Σ = 和

A 和 B = 输入变量（运算符）



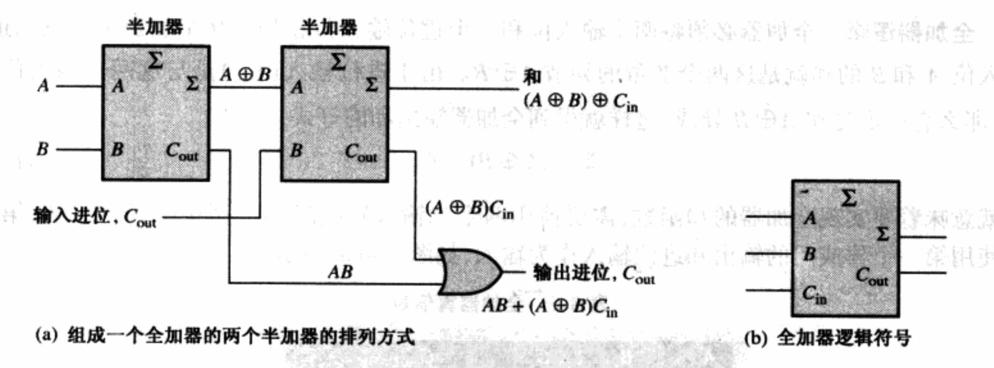
(a) 形成三个位的和所需的逻辑　　(b) 实现全加器的完整逻辑电路（每一个阴影区域是一个半加器）

图 6.4　全加器逻辑

当第一个异或门的所有输入都为 1 时，或当第二个异或门的所有输入都为 1 时，进位为 1。可以根据表 6.2 来验证这个结论。输入 A 和 B 相与再加上 $A \oplus B$ 和 C_{in} 相与产生了全加器的输出。如式(6.4)所示，两个与项相或。这个函数与和逻辑组成了一个完整的全加器电路，如图 6.4(b)所示。

$$C_{out} = AB + (A \oplus B)C_{in} \tag{6.4}$$

请注意，图 6.4(b)中有两个半加器，它们连接着图 6.5(a)所示的模块，两个输出进位相或。图 6.5(b)中的逻辑符号通常用来表示全加器。



(a) 组成一个全加器的两个半加器的排列方式 (b) 全加器逻辑符号

图 6.5 利用半加器组成全加器

例 6.1 如图 6.6 所示，这 3 个全加器中的每一个全加器的输入已给出，请确定它们的输出。

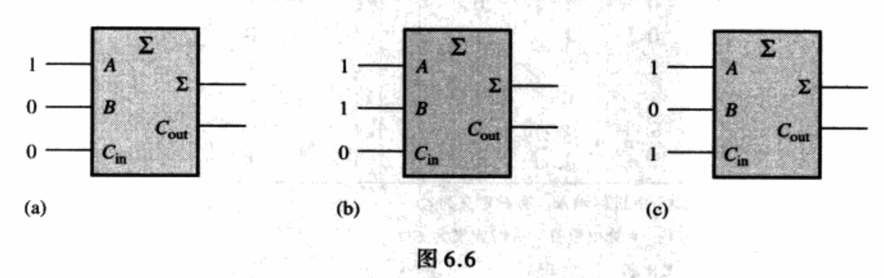


图 6.6

解：

(a)位输入为 $A=1$、$B=0$ 和 $C_{in}=0$：

$$1+0+0=1 \quad \text{没有进位}$$

所以，$\Sigma=\mathbf{1}$ 和 $C_{out}=\mathbf{0}$。

(b)位输入为 $A=1$、$B=1$ 和 $C_{in}=0$：

$$1+1+0=0 \quad \text{有进位 1}$$

所以，$\Sigma=\mathbf{0}$ 和 $C_{out}=\mathbf{1}$。

(c)位输入为 $A=1$、$B=0$ 和 $C_{in}=1$：

$$1+0+1=0 \quad \text{有进位 1}$$

所以，$\Sigma=\mathbf{0}$ 和 $C_{out}=\mathbf{1}$。

相关问题： 当 $A=1$、$B=1$、$C_{in}=1$ 时，全加器的输出是什么？

6.2 并行二进制加法器

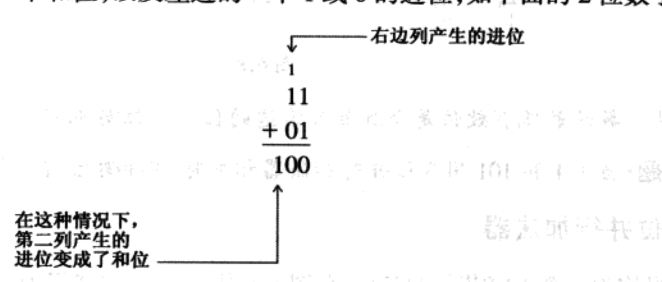
当两个或更多的全加器组合在一起时，就可以形成一个并行二进制加法器。在本节中，将会学习到这种类型的加法器的基本运算，以及与之相关联的输入和输出函数。

学完本节以后，应当能够

- 使用全加器实现一个并行二进制加法器
- 解释说明一个并行二进制加法器的运算过程

- 使用一个 4 位并行加法器的真值表
- 把 74LS283 应用于两个 4 位数字的相加运算
- 把 4 位加法器扩展为相应的 8 位或 16 位加法器

在 6.1 节已经知道,一个单一的全加器能够把两个 1 位数字和一个进位输入相加。为了把多于一位的二进制数相加,就必须使用更多的全加器。当一个二进制数与另一个二进制数相加时,每一列会产生一个和位,以及左边的一个 1 或 0 的进位,如下面的 2 位数字所示:





计算机小知识

计算机进行一次加法运算的两个数,称之为操作数。保存在 ALU(算术逻辑单元)寄存器如累加器中已存在的数称为目的操作数,与它相加的操作数称为源操作数。这两个数相加以后又被保存在累加器中。为了完成整数或小数的加法,可以分别使用 ADD 或者 FADD 指令。

如果要将两个二进制数相加,数字中的每一个位都需要一个全加器。那么 2 位数字需要两个加法器;4 位数字需要 4 个加法器;以此类推。每一个加法器的进位输出与下一个较高位的加法器的进位输入相连,如图 6.7 的 2 位加法器所示。请注意最低位有效位可以使用一个半加器,或者使用一个全加器,它的进位输入为 0(接地),因为最低有效位是没有进位输入的。

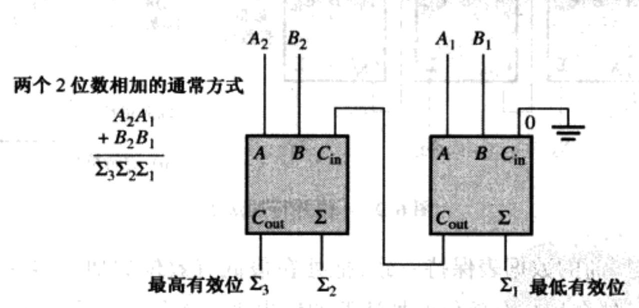


图 6.7 用两个全加器组成的一个基本 2 位并行加法器框图

在图 6.7 中,用 A_1 和 B_1 来表示这两个数字的最低有效位(LSB)。下一个较高的位用 A_2 和 B_2 来表示。3 个和位用Σ_1、Σ_2 和Σ_3 来表示。注意,全加器最左边的进位输出变成了加法结果Σ_3 中的最高有效位(MSB)。

例 6.2 在图 6.8 中,求 3 位并行加法器产生的和,并给出当二进制数 101 和 011 相加时,中间的进位。

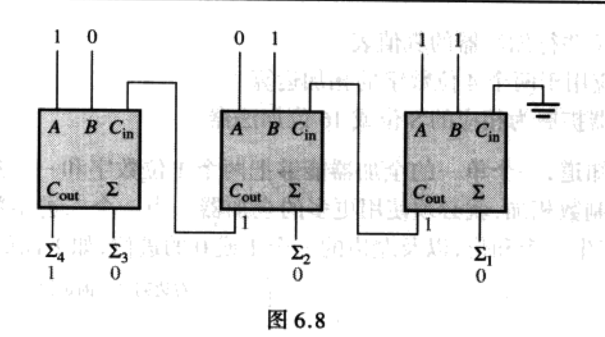


图 6.8

解:这两个数的最低有效位是全加器最右边的位。和位与中间进位如图 6.8 所示。

相关问题:当 111 和 101 用 3 位并行加法器相加时,其和输出是什么?

6.2.1　4 位并行加法器

4 位一组称为一个半字节(nibble)。在图 6.9 中,4 个全加器组成了一个基本 4 位并行加法器。如前所述,相加的每一个数中的最低有效位(A_1 和 B_1)处在最右边的全加器上,较高一级的位加在接续的较高一级的全加器上,相加的每一个数中最高有效位(A_4 和 B_4)加在最左边的全加器上。如图所示,每一个加法器的进位输出与下一个较高一级的加法器的进位输入相连。这些进位称为内部进位。

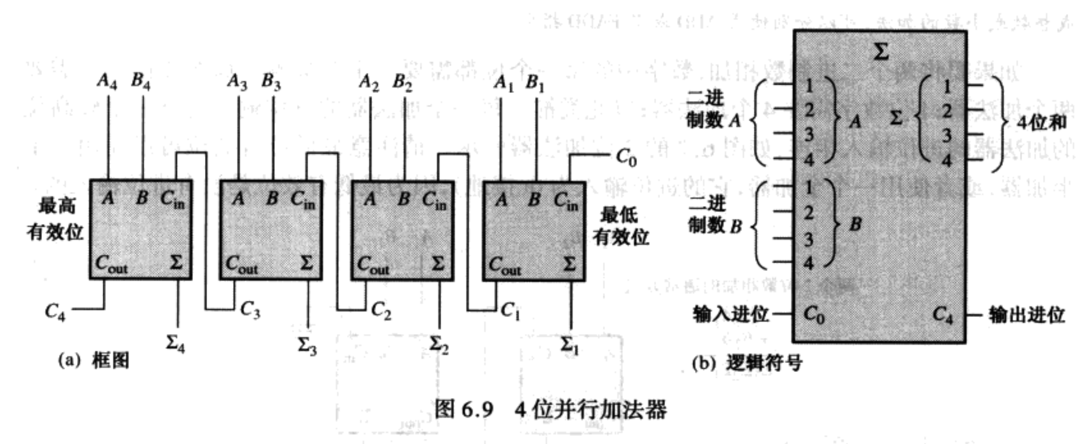


图 6.9　4 位并行加法器

为了与大多数厂商的数据表保持一致,把处在最低有效位的加法器的进位输入记为 C_0;在有 4 位的情况下,处在最高有效位的加法器的输出进位就是 C_4;最低有效位Σ_1 到最高有效位Σ_4 就是相加后的输出。逻辑符号如图 6.9(b)所示。

在一个并行加法器中,处理进位的方法有两种:异步进位加法器和超前进位加法器。串行加法器将在 6.3 节讨论。

6.2.2　4 位并行加法器的真值表

4 位加法器的真值表如表 6.3 所示。在有些数据表中,真值表称为函数表或函数真值表。下标 n 表示加法器的位,它可以是 4 位加法器的 1、2、3 或 4 位,C_{n-1}是来自前一级加法器的进

位。C_1、C_2 和 C_3 通常是内部进位。C_0 是外部进位输入，C_4 为输出。例6.3解释了表6.3的使用。

表6.3　4位并行加法器的每一级真值表

C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

例6.3　使用4位并行加法器的真值表(见表6.3)，求出下列两个4位数字的和及进位输出，如果进位输入 C_{n-1} 是0：

$$A_4A_3A_2A_1 = 1100 \quad 和 \quad B_4B_3B_2B_1 = 1100$$

解：对于 $n=1$：$A_1=0, B_1=0, C_{n-1}=0$。从真值表的第一行可得

$$\Sigma_1 = \mathbf{0} \quad 和 \quad C_1 = 0$$

对于 $n=2$：$A_2=0, B_2=0, C_{n-1}=0$。从真值表的第一行可得

$$\Sigma_2 = \mathbf{0} \quad 和 \quad C_2 = 0$$

对于 $n=3$：$A_3=1, B_3=1, C_{n-1}=0$。从真值表的第四行可得

$$\Sigma_3 = \mathbf{0} \quad 和 \quad C_3 = 1$$

对于 $n=4$：$A_4=1, B_4=1, C_{n-1}=1$。从真值表的最后一行可得

$$\Sigma_4 = \mathbf{1} \quad 和 \quad C_4 = \mathbf{1}$$

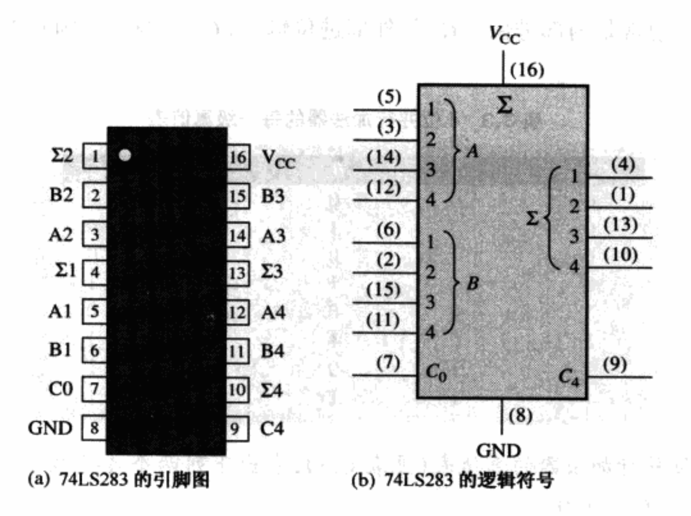
C_4 成为进位输出，1100与1100的和为11000。

相关问题：使用真值表(见表6.3)求二进制数1011和1010的和。

并行加法器74LS283

4位并行加法器实例的集成电路芯片有74LS283。74LS283的标准封装结构是：引脚16是 V_{CC}，并且引脚8接地。图6.10给出了芯片的引脚图和逻辑符号，逻辑符号圆括号里是引脚编号。这种芯片还有TTL或CMOS系列(可以查询德州仪器公司的网址：www.ti.com)。

集成电路数据表的特点　回顾前述，逻辑门有一个从输入到输出的特定的传输延迟时间 t_P。对于集成电路逻辑来说，可能有几个不同规格的 t_P。如图6.11所示，4位并行加法器有4种规格的 t_P，这只是74LS283数据表的一部分。



(a) 74LS283 的引脚图 (b) 74LS283 的逻辑符号

图 6.10 4 位并行加法器

符号	参数	限制			单位
		最小	类型	最大	
t_{PLH} t_{PHL}	传输延迟,C_0输入到任何 Σ 输出		16 15	24 24	ns
t_{PLH} t_{PHL}	传输延迟，任何 A 或 B 输入到Σ输出		15 15	24 24	ns
t_{PLH} t_{PHL}	传输延迟,C_0输入到 C_4 输出		11 11	17 22	ns
t_{PLH} t_{PHL}	传输延迟，任何 A 或 B 输入到 C_4 输出		11 12	17 17	ns

图 6.11 74LS283 的传输延迟特性

6.2.3 加法器扩展

◇ 通过级联,加法器可以扩展到更多的位。

使用两个 4 位加法器,可以扩展 4 位加法器,用以处理两个 8 位数字的加法。低 4 位加法器 C_0 的进位输入接地,因为最低有效位(LSB)是没有进位的,低 4 位加法器的进位输出与较高 4 位加法器的进位输入相连接,如图 6.12(a)所示。这个过程称之为级联。注意,在这种情况下,由于进位输出产生在第 8 位上,输出进位用 C_8 表示。低 4 位加法器是把数字中最低的或较低的 4 位有效位相加,而高 4 位加法器就是把这个 8 位数字中最高或较高的 4 位有效位相加。

类似地,如图 6.12(b)所示,还可以使用 4 个 4 位加法器的级联来处理 16 位数字的加法。注意,由于进位输出产生在第 16 位的位置上,所以输出进位用 C_{16}表示。

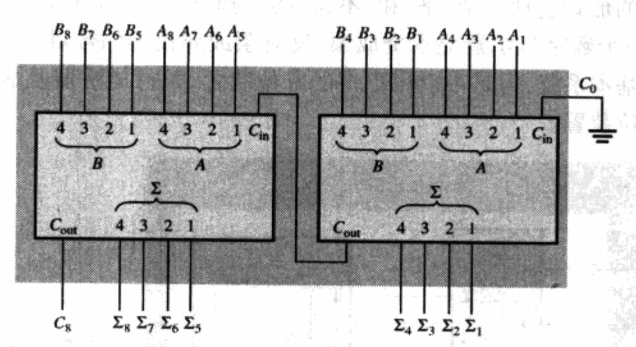
例 6.4 如何将两个 74LS283 加法器连接起来组成一个 8 位并行加法器。求下列 8 位输入数的输出位:

$$A_8A_7A_6A_5A_4A_3A_2A_1 = 10111001 \quad 和 \quad B_8B_7B_6B_5B_4B_3B_2B_1 = 10011110$$

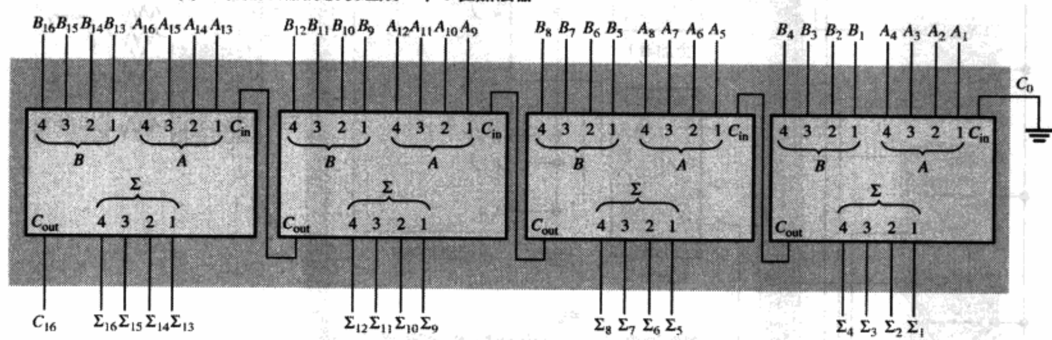
解:使用两个 74LS283 4 位并行加法器来组成一个 8 位加法器。两个 74LS283 之间的唯一连接,是这个低位加法器的进位输出(引脚 7)与高位加法器的进位输入(引脚 9)之间的连接,如图 6.13 所示。低位加法器的引脚 7 接地(没有进位输入)。

八位数的和是: $\Sigma_9 \Sigma_8 \Sigma_7 \Sigma_6 \Sigma_5 \Sigma_4 \Sigma_3 \Sigma_2 \Sigma_1 = 101010111$。

相关问题:使用 74LS283 加法器组成一个 12 位并行加法器。



(a) 4 位加法器的级联组成一个 8 位加法器



(b) 4 个 4 位加法器的级联组成一个 16 位加法器

图 6.12　加法器扩展的例子

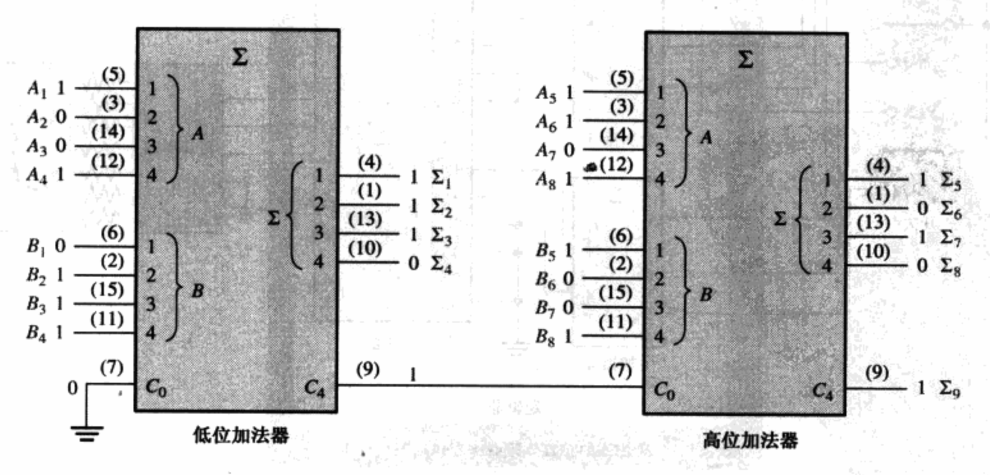


图 6.13　两个 74LS283 加法器相连组成一个 8 位加法器(括号里的数字为引脚编号)

6.2.4 应用举例

投票系统就是全加器和并行加法器的一个应用例子,它可以在投票的同时计算出"赞成"及"反对"的票数。当一群人聚集在一起,需要立即对观点(赞成或反对)进行确定;在对某个争端件事或一些其他事情进行投票时,就需要用到这种系统。

投票系统的最简单的形式包括一个"是"和"不是"的选择开关,并且在每个投票人处有一个这样的开关;同时有一个数字显示器显示赞成票、反对票或弃权票的数目。如图 6.14 所示的就是一个 6 位装置的基本系统,但是可以用更多的 6 位装置、并行加法器及显示电路,将这个 6 位装置扩展为更多位装置。

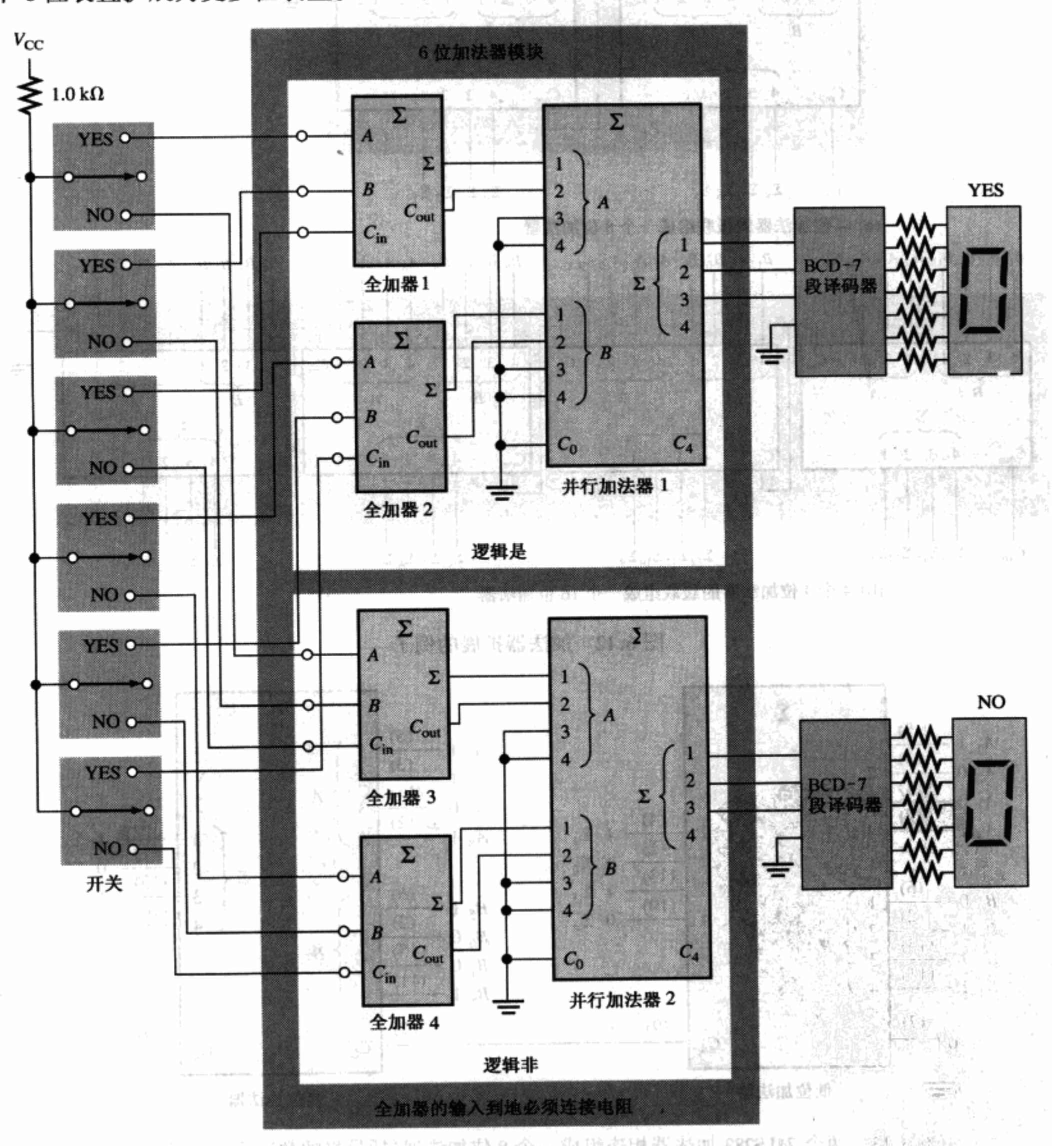


图 6.14 使用全加器和并行二进制加法器的投票系统

在图 6.14 中，每一个全加器可以产生最多 3 票的和。然后每个全加器的和及进位输出被传送到一个并行二进制加法器的两个低位。并行加法器的两个是低位输入与地连接(0)，因为在任何情况下，二进制输入都不会超过 0011(十进制数 3)。对于这种基本 6 位系统，并行加法器的输出被传送到一个 BCD-7 段译码器中。如同所提到的，如果系统扩展，必须包括更多的电路。

每个全加器的输入连接电阻到地，这样可以确保当开关处在悬空位置时，每一个输入为低电平(使用 CMOS 逻辑)。当开关拨在"是"的位置上或在"不是"的位置上时，高电平(V_{CC})加在与之相连的全加器输入。

6.3　异步进位与超前进位加法器

如上一节提到的，根据从一级到另一级的内部进位的传输方法，并行加法器可以分为两类。这两类是异步进位和超前进位。从外部输入和输出的角度来看，两种类型是相同的。区别是它们在进行加法运算时的速度不同。超前进位加法器的速度比异步进位加法器的速度要快得多。

学完本节以后，应该能够

- 讨论异步进位加法器和超前进位加法器的区别
- 陈述超前进位加法器的优点
- 定义进位的产生和延迟，解释区别
- 设计超前逻辑
- 解释为什么级联的加法器 74LS283 既有异步进位也有超前进位的特性

6.3.1　超前进位加法器

异步进位加法器中的每个全加器的进位输出连接到下一个高一级的全加器的输入(一级一个全加器)。任何一级的输出和及进位必须在上一级的进位到来后才能产生。这就造成加法过程的时间延迟，如图 6.15 所示。假设输入 *A* 和 *B* 已经到达，每个全加器的进位传输延迟是从输入进位的到达到输出进位的产生。

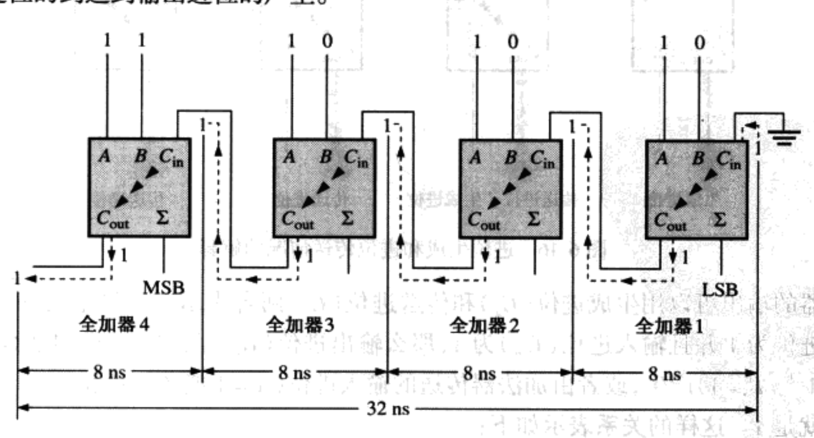


图 6.15　4 位并行异步进位加法器进位传输延迟的"最坏情况"

全加器 1(FA1)在有进位输入到达以后,才有可能产生进位输出。全加器 2(FA2)在全加器 1 产生进位输入以后,才有可能产生进位输出。全加器 3(FA3)在全加器 1 产生进位输出后,紧接着还要等全加器 2 的进位输出产生以后,才有可能产生进位输出,以此类推。如图 6.15 所示,最低有效位全加器的进位输入在产生最后的加法结果前,必须异步途经所有的全加器。所有全加器的传输延迟的积累就是“最坏情况”的加法运算时间。总的延迟时间可能不一样,取决于每个全加器产生进位的时间。如果两个数相加在两个全加器之间没有进位(0)产生,那么相加的时间仅仅是单一全加器的运算时间,即相加数据位加在输入到产生和输出的时间。

6.3.2 超前进位加法器

加法运算可以达到的速度,由于通过并行加法器的每一级的进位传输,或异步进位所需时间的缘故受到了限制。通过消除异步进位延迟,加快加法运算速度的一个方法是超前进位加法。超前进位加法器提前使用每一级的输入进位,并基于输入,通过进位生成或进位传送函数而产生进位。

进位生成 当一个输出进位由全加器在内部产生(生成)时,进位生成就发生了。当且仅在两个输入位为 1 时,就生成了一个进位。生成的进位 C_g 表示为两个输入位 A 和 B 的与运算:

$$C_g = AB \tag{6.5}$$

进位传送 当输入进位异步传送成为输出进位时,进位传送就发生了。当全加器的一个或两个输入位为 1 时,就可能传送了一个输入进位。传送的进位 C_p 表示为两个输入位的或运算:

$$C_p = A + B \tag{6.6}$$

如图 6.16 所示,即进位生成和进位传送的情况。三个箭头表示异步(传送)。

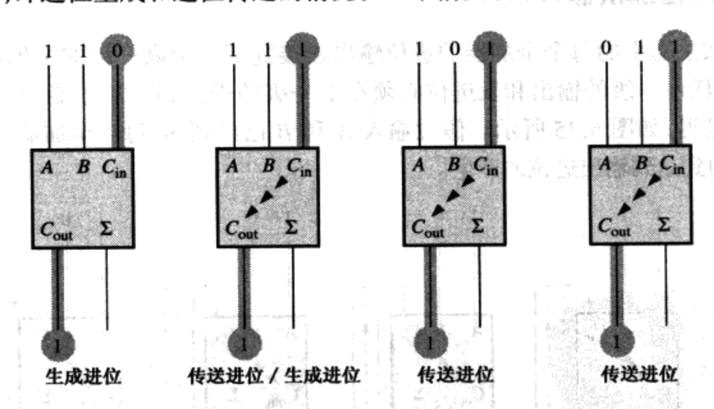


图 6.16 进位生成和进位传送情况的解释

全加器的输出可以用生成进位(C_g)和传送进位(C_p)两者表示。如果生成进位为 1,或者如果传送进位为 1 并且输入进位(C_{in})为 1,那么输出进位(C_{out})就是 1。换句话说,如果由全加器($A = 1$ 与 $B = 1$)产生,或者由加法器传送的输入进位($A = 1$ 或 $B = 1$)和 $C_{in} = 1$ 产生,那么输出进位就是 1。这样的关系表示如下:

$$C_{out} = C_g + C_pC_{in} \tag{6.7}$$

现在来看这样的概念是如何应用于一个并行加法器，它的每一级参见图 6.17 所示的 4 位加法器的例子。对于每个全加器，输出进位由生成进位(C_g)、传送进位(C_p)和输入进位(C_{in})确定。只要输入位 A、B 和最低有效位(LSB)加法器的输入进位相加，立即可以得到每一级的 C_g 和 C_p 函数，因为这两个函数仅仅由这些位确定。每一级的输入进位就是前面一级的输出进位。

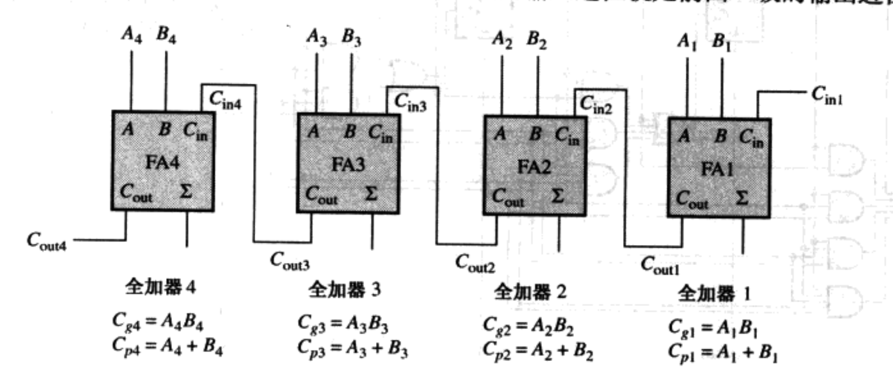


图 6.17 4 位加法器输入位的进位生成和进位传送

基于这样的分析，现在就可以设计 4 位加法器的每一级全加器的输出进位(C_{out})的表达式。

全加器 1:

$$C_{out1} = C_{g1} + C_{p1}C_{in1}$$

全加器 2:

$$\begin{aligned} C_{in2} &= C_{out1} \\ C_{out2} &= C_{g2} + C_{p2}C_{in2} = C_{g2} + C_{p2}C_{out1} = C_{g2} + C_{p2}(C_{g1} + C_{p1}C_{in1}) \\ &= C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1} \end{aligned}$$

全加器 3:

$$\begin{aligned} C_{in3} &= C_{out2} \\ C_{out3} &= C_{g3} + C_{p3}C_{in3} = C_{g3} + C_{p3}C_{out2} = C_{g3} + C_{p3}(C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1}) \\ &= C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1} \end{aligned}$$

全加器 4:

$$\begin{aligned} C_{in4} &= C_{out3} \\ C_{out4} &= C_{g4} + C_{p4}C_{in4} = C_{g4} + C_{p4}C_{out3} \\ &= C_{g4} + C_{p4}(C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1}) \\ &= C_{g4} + C_{p4}C_{g3} + C_{p4}C_{p3}C_{g2} + C_{p4}C_{p3}C_{p2}C_{g1} + C_{p4}C_{p3}C_{p2}C_{p1}C_{in1} \end{aligned}$$

注意，在所有的表达式中，每一级全加器输出进位仅由最初的输入进位(C_{in1})、那一级的函数 C_g 和 C_p 及前一级的函数 C_g 和 C_p 确定。因为每一级的函数 C_g 和 C_p 可以由这级全加器的输入 A 和 B 来表示。所有的输入进位可以立即获得(除了门的延迟)，不需要等到进位异步传送通过每一级后才得到最后结果。因此，超前进位技术加快了加法运算的速度。

如图 6.18 所示，使用逻辑门并连接全加器产生的 4 位超前加法器，也获得了 C_{out} 的表达式。

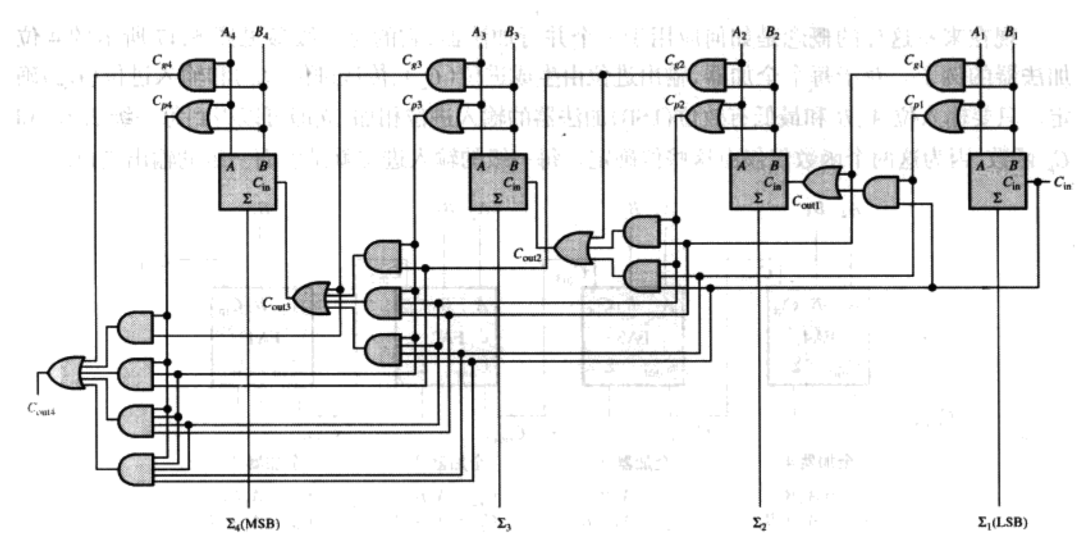


图 6.18　4 位超前加法器的逻辑图

6.3.3　超前加法器和异步进位加法器的组合

6.2 节介绍的 4 位加法器 74LS283 是一个超前加法器。当这些加法器级联起来、把处理二进制数的能力扩展到四位以上时,一个加法器的输出进位连接下一个加法器的输入进位。当两个以上的 74LS283 级联时,就在 4 位加法器中间产生了异步进位。最终的加法器实际上是一个超前和异进位的组合。每个中规模集成电路(MSI)加法器的内部运行于超前进位,当有进位在一个加法器的外部传到下一个时,异步进位的特点就起作用了。

6.4　比较器

比较器的基本功能是比较两个二进制数的大小以确定这两个数量的关系。其最简单的形式,就是用一个比较电路判断两个数字是否相等。

学完本节以后,应当能够

- 把异或门用做一个基本比较器
- 分析一个具有相等和不相等的大小比较器的内部逻辑
- 把 74HC85 比较器用于比较两个 4 位数字的数值大小
- 把 74HC85 级联,使一个比较器扩展到 8 位或更多位

6.4.1　相等

如在第 3 章中所学到的,异或门可以用做一个基本的比较器,因为如果它的两个输入位不相等时输出为 1,相等时输出为 0。图 6.19 给出了异或门用做一个 2 位比较器的情况。

为了比较两个 2 位二进制数的大小,需要更多的异或门。门 G_1 用来比较这两个数的两个最低有效位(LSB),门 G_2 用来比较两个最高有效位(MSB),如图 6.20 所示。如果两个数字是

相等的，它们所对应的位也是相同的，那么，每个异或门的输出为 0。如果对应的两个位不相等，则异或门的输出为 1。

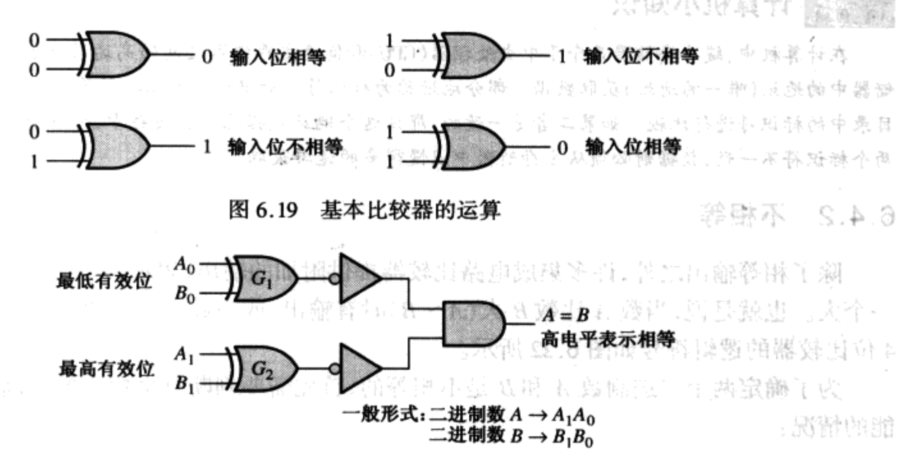


图 6.20　比较两个相等的 2 位数字的逻辑图

◇ 一个比较器用来判断两个二进制数是否相等。

为了产生一个单一的输出来表示两个数字相等还是不相等，需要使用两个反相器和一个与门，如图 6.20 所示。每个异或门的输出被反相，加到与门的输入。当每一个异或门的两个输入相等时，与门的两个输入为 1，因此与门的输出为 1。当两个数不相等时，相应的一对位或两对位不相等，这时与门中至少有一个输入为 0，与门的输出为 0。因此，与门的输出表示两个数的相等(1)或不相等(0)。

例 6.5 解释了这种运算的两种具体情况。同或门符号代替了异或门和反相器。

例 6.5　在图 6.21 中，下列每个二进制数加到比较器的输入，求下列逻辑电平通过电路时的输出。

(a) 10 和 10　　　(b) 11 和 10

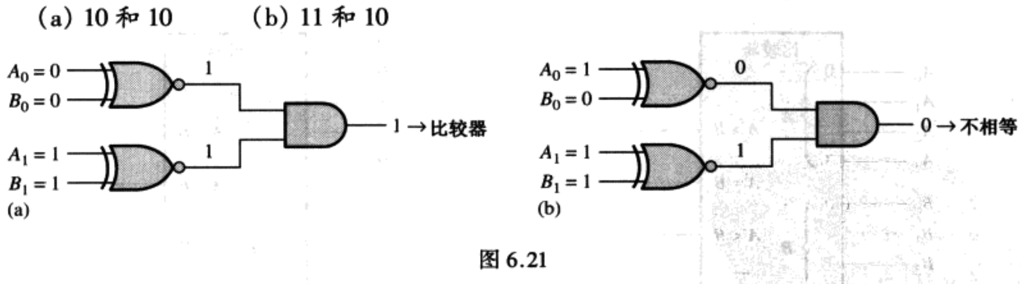


图 6.21

解：

(a)当输入为 10 和 10 时，输出为 **1**，如图 6.21(a)所示。

(b)当输入为 11 和 10 时，输出为 **0**，如图 6.21(b)所示。

相关问题：当二进制输入为 01 和 10 时，重复这个过程。

如在第 3 章所知道的，这个基本比较器可以扩展到任何位数。当这两个数自身相等时，与门决定了它们所对应的位必定相等。



计算机小知识

在计算机中,缓冲存储器是介于中央处理器(CPU)和低速主存储器之间的高速存储器。CPU 通过它在存储器中的地址(唯一的地址)获取数据。部分地址称为标识符。标识符地址比较器将 CPU 中的标识符与缓存目录中的标识符进行比较。如果二者是一致的,那么这个地址数据已经在缓存中,可以很快地获取。如果这两个标识符不一致,数据则必须从主存储器中以慢得多的速率获取。

6.4.2 不相等

除了相等输出之外,许多集成电路比较器提供附加的输出,表示两个相比较的二进制数哪一个大。也就是说,当数 A 比数 B 大($A>B$)时有输出,或当数 A 比数 B 小($A<B$)时有输出。4 位比较器的逻辑符号如图 6.22 所示。

为了确定两个二进制数 A 和 B 是不相等的,首先需要判断每个数字的最高位。下面是可能的情况:

1. 如果 $A_3=1$ 和 $B_3=0$,数 A 大于数 B;
2. 如果 $A_3=0$ 和 $B_3=1$,数 A 小于数 B;
3. 如果 $A_3=B_3$,这时必须判断下一个较低位的不相等性。

这三个运算对于上述数字的每个位都是有效的。比较器运算的大致过程就是从最高有效位(MSB)开始,检查这一个位的不等性。当找到了这个不等性时,这两个数字之间的关系也就确立了,其他较低位的大小关系可以忽略,因为较低位可能会有与之相反的情况发生;最高位的结果必须优先考虑。

例 6.6 如图 6.23 所示,比较器的输入数字已经给出,求当 $A=B$、$A>B$ 和 $A<B$ 时的输出。

解:数字 A 的输入为 0110,数字 B 的输入为 0011。$A>B$ 的输出为高电平,其他输出为低电平。

相关问题:当 $A_3A_2A_1A_0=1001$ 和 $B_3B_2B_1B_0=1010$ 时,求比较器的输出?

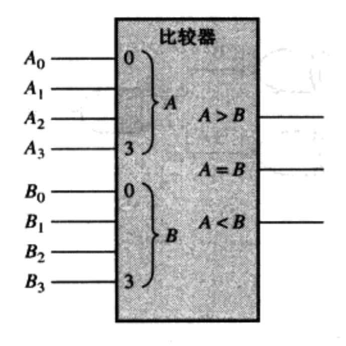


图 6.22 具有不相等输出的 4 位比较器的逻辑符号

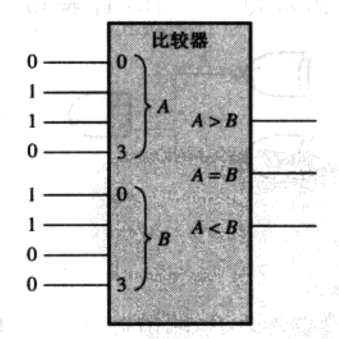


图 6.23

74HC85 4 位大小比较器

74HC85 是一个比较器,在其他集成电路系列中也可以找到它。74HC85 比较器的引脚图

和逻辑符号如图6.24所示。注意,这种比较器除了含有前面介绍过的常用比较器的所有输入及输出之外,还有3个级联输入:$A<B$、$A=B$和$A>B$。这些输入允许几个比较器级联起来,用来比较大于4位数字的数。为了扩展比较器,低位比较器的$A<B$、$A=B$和$A>B$输出与对应的下一个较高位比较器的级联输入相连接。最低位比较器在$A=B$输入端口必须接高电平,在$A<B$和$A>B$输入端口必须接低电平。这种芯片在其他的CMOS或TTL系列中也有。请查阅德州仪器公司的网址:www.ti.com。

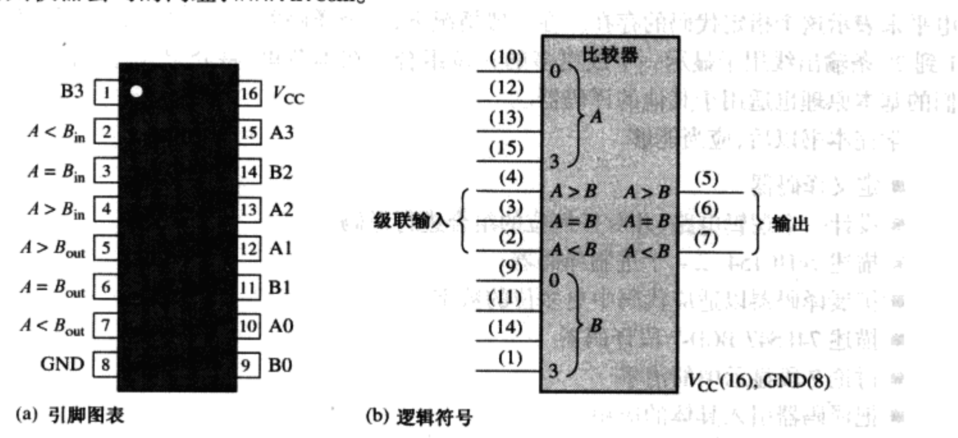


图6.24　74HC85 4位大小比较器的引脚图表和逻辑符号(引脚编号在圆括号内)

例6.7　使用74HC85比较器比较两个8位数的数值大小。给出比较器的恰当连接。

解:要比较两个8位数需要两个74HC85芯片。如图6.25所示,两个芯片以级联的方式连接。

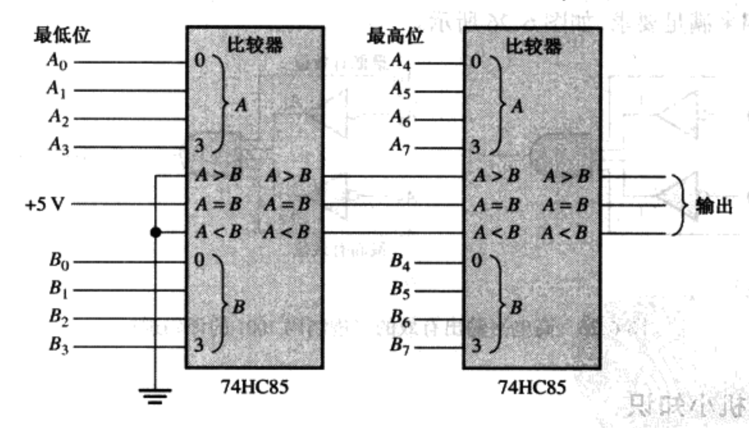


图6.25　使用两个74HC85的8位大小比较器

相关问题:将图6.25中的电路进行扩展,使其成为一个16位比较器。

相关提示

多数CMOS器件都含有一个保护电路,用于避免由于高电压或很强的静态电场所造成的损坏。然而,还是必须小心,避免接触任何高于最大额定值的电压。为了正确使用,输入

和输出的电压值应该保持在接地和电源电压 V_{CC}之间。此外,记住未被使用的输入端口必须连接相应的逻辑电平(地或 V_{CC})。未被使用的输出端口可以断开。

6.5 译码器

译码器是一个数字电路,用以检测其输入端位的指定组合(代码)是否存在,由指定的输出电平来表示这个指定代码的存在。在一般情况下,一个译码器有 n 条输入线,用于处理 n 位,1 到 2^n 条输出线用于显示一个或更多的 n 位组合。在本节里,将介绍几种不同的译码器,它们的基本原理也适用于其他的译码器。

学完本书以后,应当能够

- 定义译码器
- 设计一个逻辑电路,并对任何位的组合进行译码
- 描述 74HC154 二-十进制译码器
- 扩展译码器以适应代码中更多位的数字
- 描述 74LS47 BCD-7 段译码器
- 讨论 7 段显示中的消零
- 把译码器引入具体的应用

6.5.1 基本二进制译码器

假设一个二进制数 1001 出现在一个数字电路的输入中,需要对它进行确认。在这个基本译码电路中,需要使用一个与门,因为只有当所有的输入都为高电平时,它才产生一个高电平。因此,当二进制数以 1001 出现时,就必须确定与门的所有输入都为高电平;这可以通过把中间两个位(0)反相来满足要求,如图 6.26 所示。

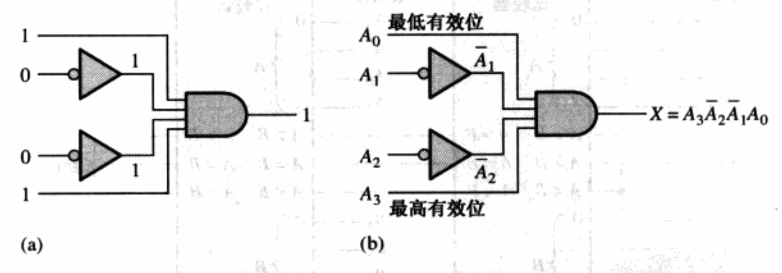


图 6.26 高电平输出有效的二进制码 1001 的译码逻辑



计算机小知识

指令告诉计算机该进行什么运算。指令以机器码(1 和 0)的形式存在,为了使计算机执行指令,需要对指令译码。指令译码是指令流水线中一步,过程如下:从存储器读取指令(取指令),指令被译码,从存储器读取操作数(取操作数),执行指令,结果写回存储器。基本上,流水线就是使当前的指令完成后,下一条指令开始进行。

对逻辑方程进行解释。可以验证,当输入 $A_0=1$、$A_1=0$、$A_2=0$ 和 $A_3=1$ 时,输出为 1,其他情况输出为 0。其中 A_0 是最低有效位,A_3 是最高有效位。本书在一个二进制数或其他有权码的表示方法中,最低有效位(LSB)是水平方向最右边的位,或者垂直方向最上面的位,除非具体说明。

在图 6.26 中，如果用一个与非门来代替这个与门，一个低电平输出将表示相应二进制代码的输入，在此情况下，二进制代码仍然是 1001。

例 6.8 求出对二进制数 1011 译码、译码输出为高电平的译码逻辑电路。

解：在所求的二进制数中，只要对以 0 出现的变量取反就可以得到译码函数，如下所示：

$$X = A_3\overline{A}_2A_1A_0 \qquad (1011)$$

把原变量 A_0、A_1 和 A_3 和与门的输入直接相连，再把变量 A_2 取反以后加到与门的输入，就实现了以上的函数。译码逻辑电路如图 6.27 所示。

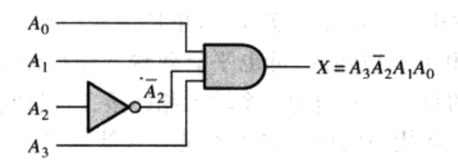


图 6.27 当输入为 1011 时产生一个高电平输出的译码逻辑电路

相关问题：将这个译码逻辑电路扩展，用来检测二进制代码 10010，并产生一个低电平有效输出。

6.5.2 4 位译码器

为了对所有的 4 位组合进行译码，需要 16 个译码门（$2^4 = 16$）。这种译码器通常称为 4 线 - 16 线译码器，因为有 4 个输入和 16 个输出，或者称为 16 选 1 译码器，因为给出的任何一个输入码，在 16 个输出里只有一个是有效的。表 6.4 给出 16 个二进制代码和与之相对应的译码函数。

表 6.4 低电平有效输出的 4 线 - 16 线（16 选 1）译码器的译码函数和真值表

十进制数	二进制输入				译码函数	输出															
	A_3	A_2	A_1	A_0		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

如果对于每个译码的数,需要一个高电平输出,整个译码器可以由与非门和反相器来实现。为了对 16 个二进制码进行译码,需要 16 个与非门(与门可以用来产生高电平有效输出)。

图 6.28 是一个低电平有效输出的 4 线 - 16 线(16 选 1)译码器的逻辑符号。BIN/DEC 表示的是一个二进制输入产生相对应的十进制有效输出。输入标号为 8、4、2 和 1 的输入代表的是输入位的二进制权($2^3 2^2 2^1 2^0$)。

74HC154 16 选 1 译码器

74HC154 是集成电路译码器的一个很好的例子。逻辑符号如图 6.29 所示。这个器件提供了一个使能函数(EN),它由一个或非门实现,这里把或非门作为一个非 - 与门来使用。每个片选输入$\overline{CS_1}$和$\overline{CS_2}$为低电平有效输入,低电平有效输入使得使能门输出(EN)高电平。这个使能门输出和译码器中的每个与非门的一个输入相连接,所以使能门输出必须是高电平,这样才能使与非门工作。如果使能门的两个输入不是有效低电平,这时无论 4 个输入变量 A_0、A_1、A_2 和 A_3 是什么值,译码器的所有 16 个输出(Y)全部为高电平。这种芯片在其他的 CMOS 或 TTL 系列中也有。请查阅德州仪器公司的网址:www.ti.com。

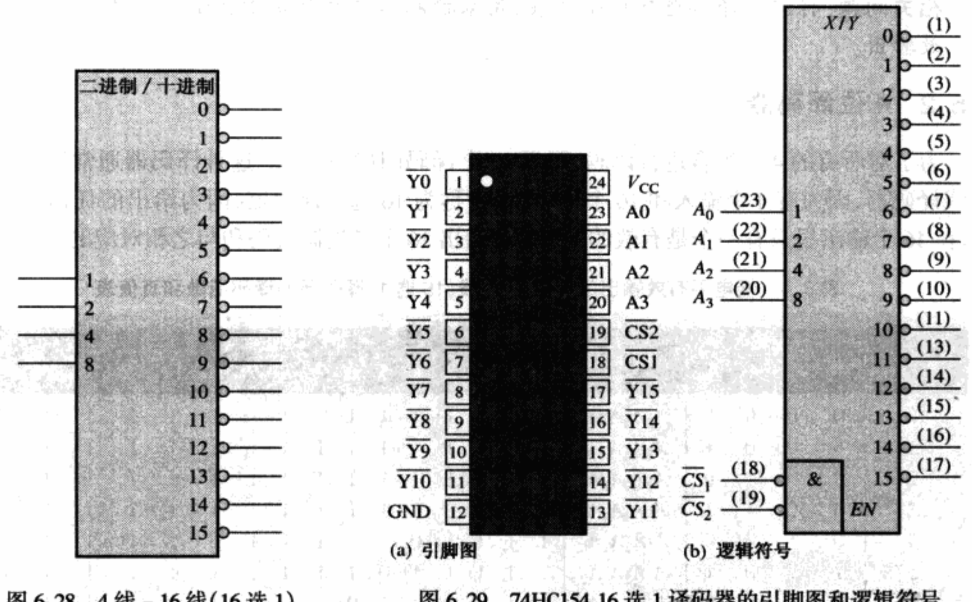


图 6.28 4 线 - 16 线(16 选 1)译码器的逻辑符号

图 6.29 74HC154 16 选 1 译码器的引脚图和逻辑符号

例 6.9 某些实际应用需要对一个 5 位数译码。利用 74HC154 译码器实现这个逻辑。二进制数用 $A_4A_3A_2A_1A_0$ 形式表示。

解: 由于 74HC154 只能处理 4 位数,那么必须使用两个译码器对 5 位数进行译码。第 5 位 A_4 和其中一个译码器的片选输入$\overline{CS_1}$和$\overline{CS_2}$相连接,$\overline{A_4}$与另一个译码器的输入$\overline{CS_1}$和$\overline{CS_2}$相连接,如图 6.30 所示。当十进制数为 15 或更小时,$A_4=0$,低位译码器工作,高位译码器不工作。当十进制数大于 15 时,$A_4=1$ 而$\overline{A_4}=0$,高位译码器工作,低位译码器不工作。

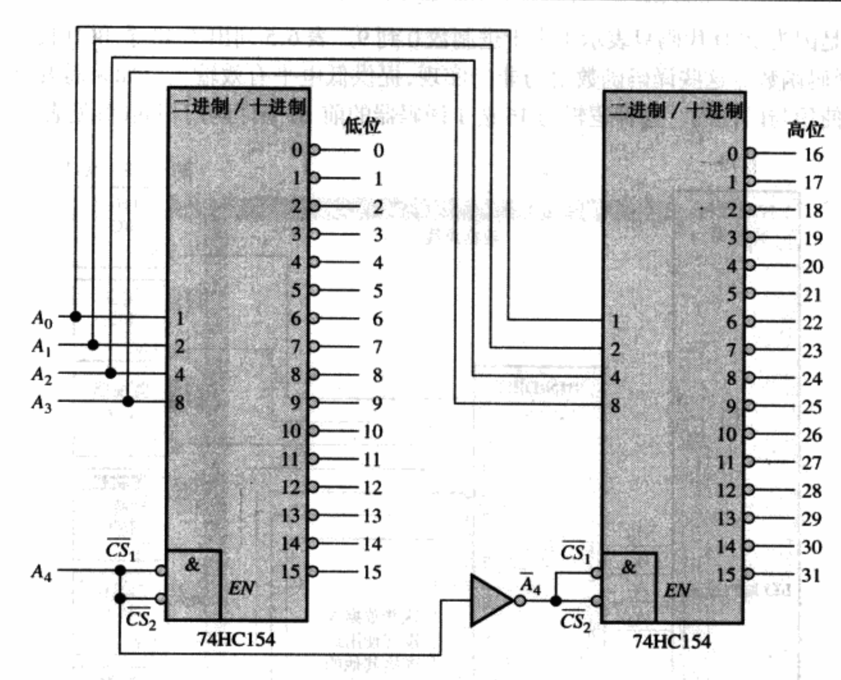


图 6.30 使用 74HC154 的 5 位译码器

相关问题：如图 6.30，求二进制有效输入 10110 的输出。

6.5.3 应用举例

译码器在许多应用场合中使用。在图 6.31 描绘的通常电路图中，给出了计算机中用于选择输入/输出的例子。计算机必须和各种外部称为周边的设备通信，可以通过输入/输出(I/O)端口发送和/或接收数据来完成。这些外部设备包括打印机、调制解调器、扫描仪、外部磁盘驱动、键盘、视频监视器和其他计算机。如图 6.31 所示，计算机用译码器选择 I/O 端口，这样可以向一个具体的外部设备传送数据或从它接收数据。

每一个输入/输出端口都有唯一识别的数字，称为地址。当计算机需要和一台特定的设备通信时，它发出相应的地址码，也就是和这个特定设备连接的输入/输出(I/O)端口地址。这个二进制端口地址被译码器译码，然后译码器输出所需的电平，使输入/输出(I/O)端口开通。

如图 6.31 所示，二进制数据在计算机内部的数据总线上传输，数据总线是一组并行的线。例如，一组 8 位总线由并行的 8 条线组成，每次可以传输一个数据字节。数据总线和所有输入/输出端口连接；但是，任何进入或出来的数据都必须通过地址译码器的译码，使得端口开通才能够传送。

6.5.4 BCD－十进制译码器

BCD－十进制译码器把每一个 BCD 代码(8421 代码)转换成 10 个十进制数中的一个数所表示的输出。它经常被称为 4 线－10 线译码器或 10 选 1 译码器。

实现此译码器的方法与前面所讲到的 16 选 1 译码器是相同的，唯一的不同就是只需要10 个

译码门,这是因为 BCD 代码只表示十个十进制数 0 到 9。表 6.5 列出了 10 个 BCD 代码及与它们相对应的译码函数。这些译码函数由与非门实现,提供低电平有效输出。如果需要高电平有效输出,那么使用与门译码。这种逻辑与 16 选 1 译码器的前 10 个译码门相同(参见表 6.4)。

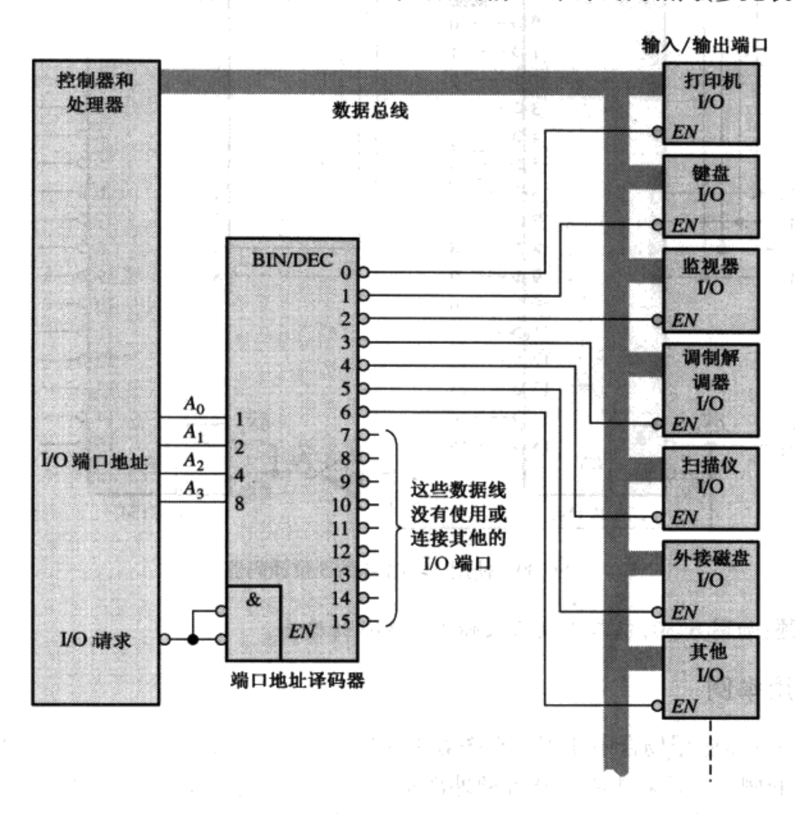


图 6.31 只有 4 条地址线和一个端口地址译码器的简单计算机 I/O 端口系统

表 6.5 BCD 译码函数

十进制数	BCD 码				译码函数
	A_3	A_2	A_1	A_0	
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$

例 6.10　74HC42 是一片集成电路 BCD－十进制译码芯片。它的逻辑符号如图 6.32 所示。如果图 6.33(a)的输入波形加在 74HC42 的输入中，请给出其输出波形。

解：如图 6.33(b)所示的输出波形。由图可以知道，输入按照 BCD 码 0 到 9 的顺序变化。时序图中的输出波形给出了十进制值输出的顺序。

相关问题：当 BCD 码的输入序列为下列十进制数：0,2,4,6,8,1,3,5,9 时，构建一个时序图，显示相应的输入和输出波形。

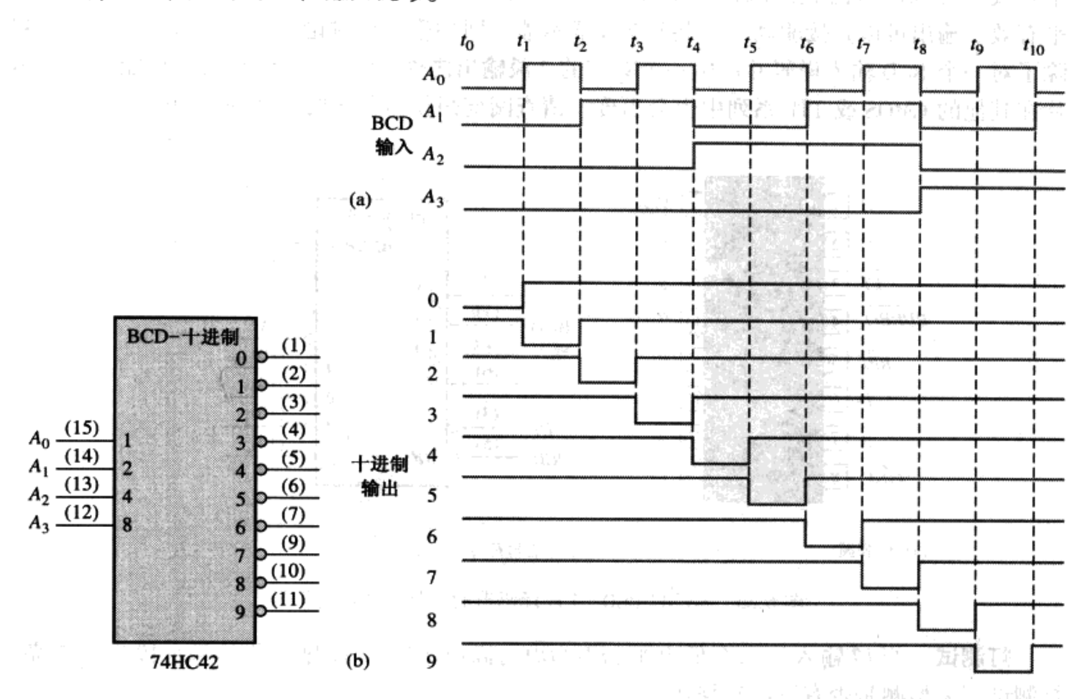


图 6.32　74HC42 BCD－十进制译码器

图 6.33

6.5.5　BCD－7 段译码器

BCD－7 段译码器的输入接受输入 BCD 代码，并产生一个用来驱动 7 段显示器的输出，显示器产生一个十进制读数。一个基本 7 段译码器的逻辑图表如图 6.34 所示。

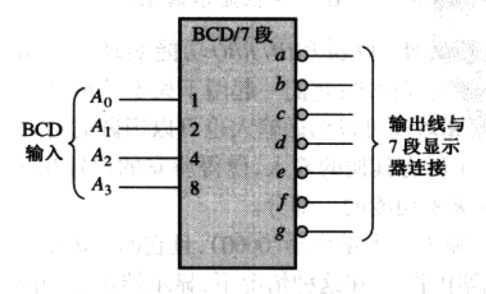


图 6.34　低电平有效输出的 BCD－7 段译码器/驱动器的逻辑符号

74LS47 BCD – 7 段译码器/驱动器

74LS47 是集成电路器件的一个例子,它对一个 BCD 输入进行译码,并驱动 7 段显示器。除了译码和段驱动功能之外,74LS47 还有另外几个附加的功能,如图 6.35 所示逻辑符号中的 $\overline{LT}$、$\overline{RBI}$、$\overline{BI}/\overline{RBO}$的功能。图中这些逻辑符号上的小圆圈,表示所有的输出(a 到 g)都是低电平有效,同样$\overline{LT}$(灯测试)、$\overline{RBI}$(异步灭零输入)和$\overline{BI}/\overline{RBO}$(灭灯输入/异步灭零输出)也是低电平有效。输出可以直接驱动一个共阳 7 段显示器。回顾第 4 章讨论过的 7 段显示器。74LS47 除了对一个 BCD 输入译码并产生一个响应的 7 段输出之外,还有灯测试和灭零功能。这种芯片在其他的 CMOS 或 TTL 系列中也会出现。请查阅德州仪器公司的网址:www.ti.com。

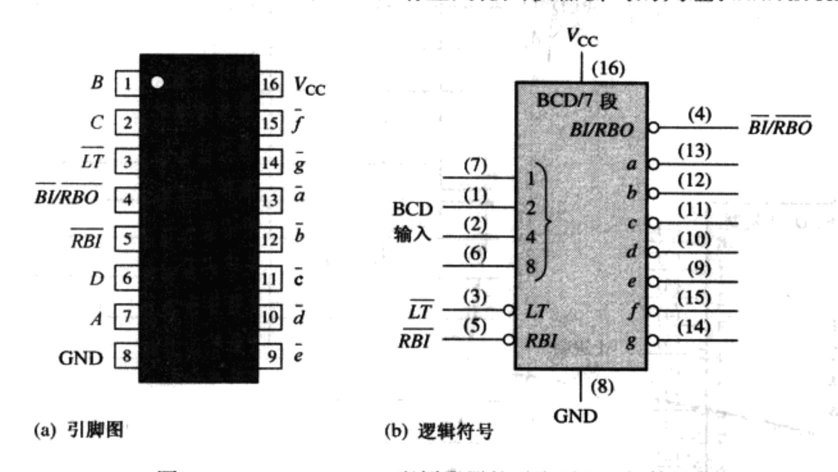


图 6.35 74LS47 BCD – 7 段译码器的引脚图和逻辑符号

灯测试 当$\overline{LT}$输入为一个低电平、$\overline{BI}/\overline{RBO}$为高电平时,7 段显示器的每一段都被点亮。灯测试用来检测是否有段已经烧坏。

灭零 灭零用于取消多位数中不必要的 0 的显示。例如,在 6 位数字显示器中,如果 0 没有熄灭,数字 6.4 可能被显示为 006.400。把数字前面的 0 熄灭称为头部灭零,把数字尾端的 0 熄灭称为尾部灭零。记住,只是把不需要的零熄灭。由于有了灭零,数字 030.080 将显示为 30.08(需要的零仍然保留)。

◇ 灭零使得一个数头部或尾部的 0 不会在显示器上显示。

在 74LS47 中,灭零是通过使用$\overline{RBI}$和$\overline{BI}/\overline{RBO}$功能实现的。$\overline{RBI}$是 74LS47 的异步灭零输入,而$\overline{RBO}$是 74LS47 的异步灭零输出;它们一起用于灭零。$\overline{BI}$是灭灯输入,它与$\overline{RBO}$共享同样的引脚;换句话说,引脚$\overline{BI}/\overline{RBO}$既可以用做输入也可以用做输出。当把它看做输入$\overline{BI}$(灭灯输入)且为低电平时,它优先于所有其他的输入,使得所有的段输出为高电平(无效状态)。灭灯输入($\overline{BI}$)功能不属于芯片灭零功能的一部分。

如果在译码器的 BCD 输入一个零代码(0000),且它的$\overline{RBI}$为低电平,那么这个译码器的所有段输出都是无效状态(高电平)。在这种情况下,显示器熄灭,并在$\overline{RBO}$产生一个低电平。

在图 6.36(a)中,给出了一个整数的头部灭零逻辑图。当译码器的 BCD 输入为 0 时,最高

有效位(最左边)的输出0总是被熄灭,因为最高有效位译码器的$\overline{RBI}$接地为低电平。每个译码器的$\overline{RBO}$连接到低一级译码器的$\overline{RBI}$上,这样从第一个非零数字开始,左边的零全部熄灭。例如,图6.36(a)中两个最高位的数字是零,因此全都熄灭。剩下的两个数字3和9是显示的。

如图6.36(b)所示,给出一个小数的尾部灭零的逻辑图表。当译码器的BCD输入为0时,最低有效位(最右边)的输出0总是被熄灭,因为最低有效位译码器的$\overline{RBI}$接地为低电平。每个译码器的$\overline{RBO}$连接到高一级译码器的$\overline{RBI}$上,这样从第一个非零数字开始,右边的零全部熄灭。例如,图6.36(b)中两个最低位的数字是零,因此全都熄灭。剩下的两个数字5和7是显示的。为了在同一显示器上使头部和尾部灭零合在一起,以及能够显示十进制数的小数点,还需要附加一些逻辑功能。

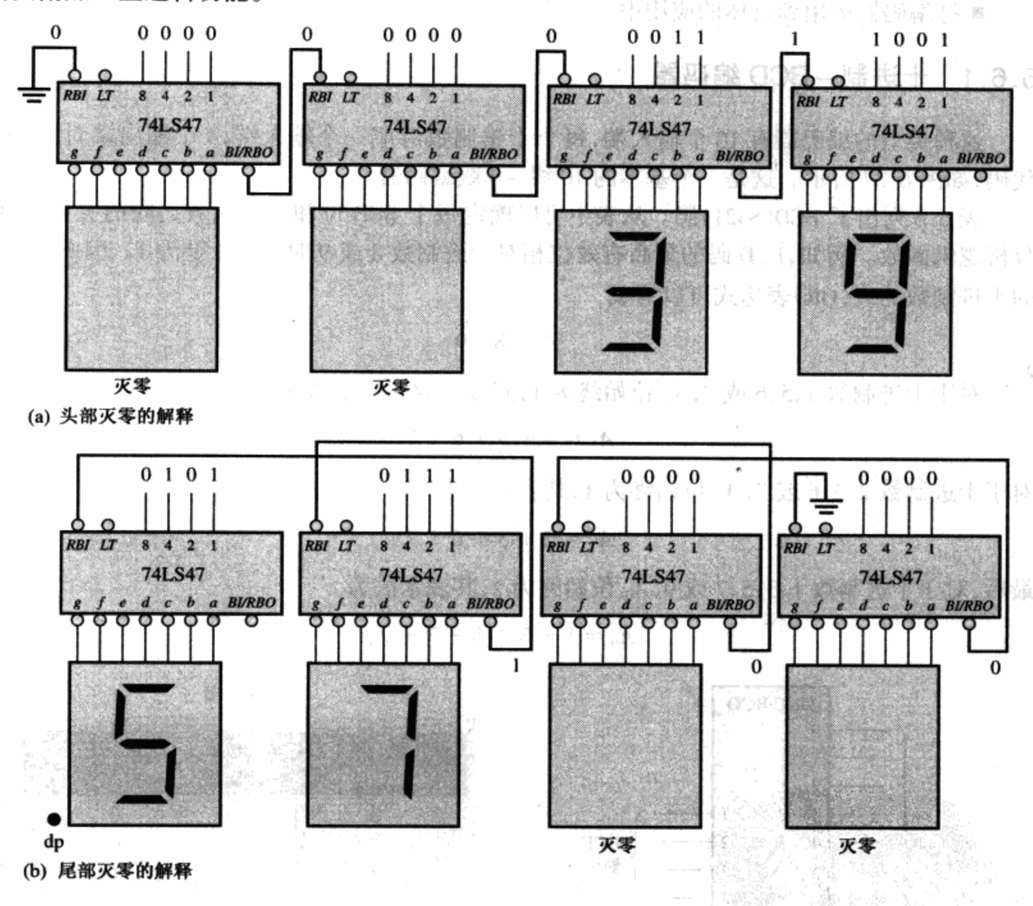


图6.36 使用74LS47 BCD-7段译码器/驱动器的灭零举例

6.6 编码器

编码器是一个逻辑电路的组合,基本上是译码器功能的反操作。编码器接收若干输入中的一个有效电平,每个输入表示一个数,例如十进制数或八进制数,并且把这个数转换为代码

输出,如 BCD 或二进制码。编码器可以设计用来对各种不同的符号和字母的字符进行编码。把熟悉的符号或数字转换成一种代码形式的过程,称之为编码。

学完本节以后,应当能够

- 确定一个十进制编码器的逻辑
- 解释编码器优先功能的用途
- 描述 74HC147 十进制 - BCD 优先编码器
- 描述 74LS148 八进制 - 二进制优先编码器
- 扩展编码器
- 把编码器运用到具体的应用中

6.6.1 十进制 - BCD 编码器

这种类型的编码器有 10 个输入端,每个十进制数对应一个输入端;4 个输出端对应 BCD 代码,如图 6.37 所示。这是一个基本的 10 线 - 4 线编码器。

表 6.6 列出了 BCD(8421)码。从表中可以确定每个 BCD 位和十进制数之间的关系,以便分析逻辑函数。例如,BCD 码的最高有效位相对十进制数 8 或 9 时,A_3 总是为 1。因此,位 A_3 的十进制数的或(OR)表达式可以写成

$$A_3 = 8 + 9$$

对于十进制数 4、5、6 或 7,A_2 位始终为 1,其或函数的表示法为

$$A_2 = 4 + 5 + 6 + 7$$

对于十进制数 2、3、6 或 7,A_1 位始终为 1,其表示法为

$$A_1 = 2 + 3 + 6 + 7$$

最后,对于十进制数 1、3、5、7 或 9,A_0 位始终为 1,其表示法为

$$A_0 = 1 + 3 + 5 + 7 + 9$$

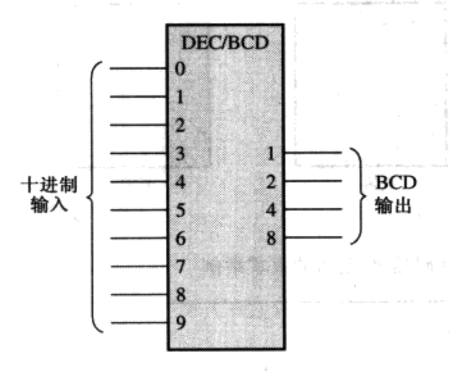
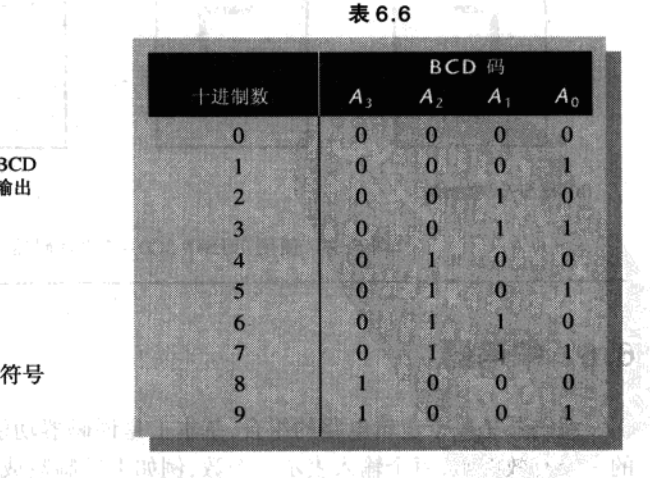


图 6.37 十进制 - BCD 编码器的逻辑符号

表 6.6



十进制数	BCD 码 A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

现在,使用刚刚推导出的这个逻辑表示法来实现这个逻辑电路,它把每一个十进制数转换成 BCD 代码。这实现起来很简单,把相对应的十进制数输入线相或以形成每个 BCD 输出。如图 6.38 所示,从这些表达式获得一个基本编码逻辑电路。

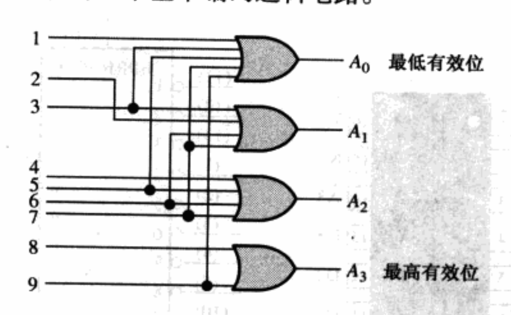


图 6.38 十进制 - BCD 编码器的逻辑图。0 输入不需要,因为当输入都是低电平时,BCD 输出也全为低电平

如图 6.38 所示的这个电路的基本操作如下:当一条十进制输入线路为高电平时,在 BCD 输出线上会产生相应的输出电平。例如,如果输入线路 9 为高电平(假设所有其他输入线为低电平),在这种条件下,输出端 A_0 和 A_3 产生高电平,A_1 和 A_2 产生低电平,这就是十进制数 9 的 BCD 码(1001)。



计算机小知识

一个汇编程序可以看做是一个软件编码器,因为它对一个程序写出的助记指令进行编译,通过把每个助记符转换成计算机可以识别的机器码指令(一系列 0 和 1),实现了把助记指令转换成可执行代码的过程。微处理器的助记指令的例子是:ADD(加),MOV(传送数据),MUL(相乘),XOR,JMP(跳转),OUT(输出到端口)。

十进制 - BCD 优先编码器 这种类型的编码器具有的基本编码功能和前面讨论的相同。一个优先编码器还提供了附加的灵活特性,即它可以应用在需要优先确定的场合。优先权功能意味着编码器将按照最高位十进制数的有效输入产生一个 BCD 输出,而不考虑任何其他低位的有效输入。例如,如果 6 和 3 都为有效输入,BCD 输出为 0110(表示十进制数 6)。

74HC147 十进制 - BCD 编码器

74HC147 为优先编码器,对于十进制数 1 到 9,输入(0)低电平有效,且为 BCD 低电平有效输出,如图 6.39 中的逻辑符号所示。当输入都为无效时,则会产生一个 BCD 零输出。括号中是这个芯片的引脚数。这种芯片在其他的 CMOS 或 TTL 系列中也有。请查阅德州仪器公司的网址:www.ti.com。

74LS148 8 线 - 3 线编码器

74LS148 是一个优先编码器,它有 8 个低电平有效输入端和 3 个二进制低电平有效输出端,如图 6.40 所示。这种芯片可以用于把八进制输入(回顾从 0 到 7 的八进制数)转换为 3 位二进制代码。为了使芯片工作,*EI*(使能输入)必须为低电平。还有用于扩展功能的 *EO*(使能输出)和 *GS* 输出。当 *EI* 为低电平且所有输入(0~7)都为无效时,*EO* 为低电平。当 *EI* 为低

电平且任何一个输入为有效输入时，*GS* 为低电平。这种芯片在其他的 CMOS 或 TTL 系列中也有。请查阅德州仪器公司的网址：www.ti.com。

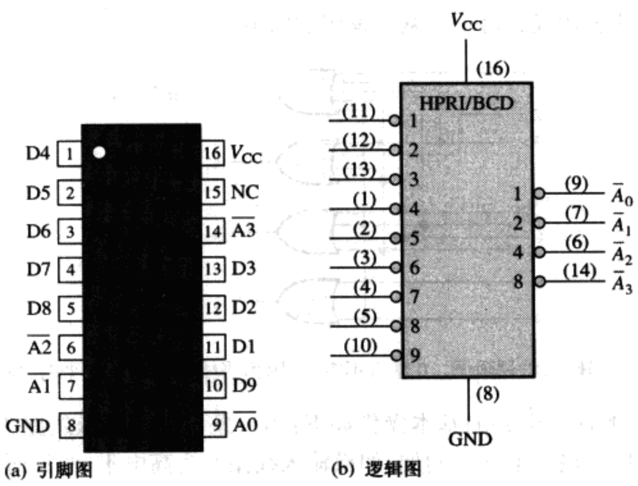


图 6.39　74HC147 十进制 - BCD 优先编码器的引脚图和逻辑符号(HPRI 表示最高值输入优先)

如图 6.41 所示，通过把高位编码器的 *EO* 连接到低位编码器的 *EI*，把相对应的输出连接到非 - 或门，就可以把 74LS148 扩展成一个 16 线 - 4 线编码器。*EO* 用做第 4 位最高有效位。这种特定的电路产生的结果是 4 位二进制高电平有效输出。

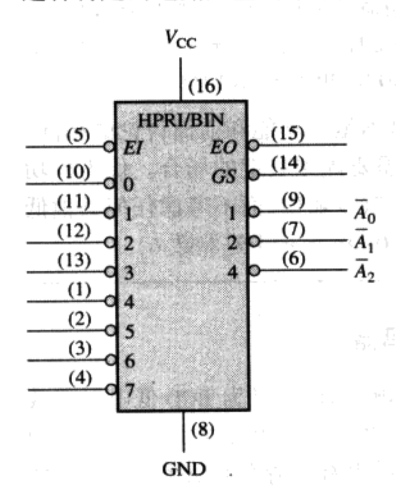


图 6.40　74LS148 8 线 - 3 线编码器的逻辑图（HPRI 表示最高值输入优先）

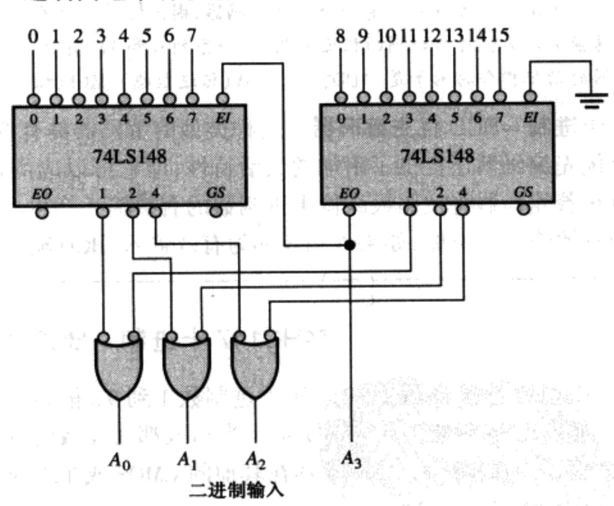


图 6.41　使用 74LS148 和外部逻辑电路的 16 线 - 4 线编码器

例 6.11　如图 6.39 所示，如果 74HC147 中引脚 1、4 和 13 为低电平时，试指出这 4 个输出的状态。所有其他输入为高电平。

解: 引脚 4 为十进制数输入的最高位,它为低电平,代表十进制数 7。因此,输出电平表示的是十进制数 7 的 BCD 代码,其中$\overline{A_0}$为最低有效位,$\overline{A_3}$为最高有效位。输出$\overline{A_0}$为低电平,$\overline{A_1}$为低电平,$\overline{A_2}$为低电平,$\overline{A_3}$为高电平。

相关问题: 如果所有的输入都为低电平时,74HC147 的输出是什么?如果所有的输入都为高电平时,其输出是什么?

6.6.2　应用举例

键盘编码器是一个典型的应用例子。例如,计算机键盘上的 10 个十进制数必须编码以便让逻辑电路处理。当按下其中一个键时,十进制数会被编码变成相应的 BCD 代码。如图 6.42 所示,使用一片 74HC147 优先编码器,组成一个简单的键盘编码器。键盘上的按键为 10 个按钮,每个按钮都有一个上拉电阻(pull-up resistor)接电压 + V。当按键未被接下时,这些上拉电阻可以确保线路为高电平。当按键被按下时,线路与地相连接,低电平加在相应的编码器输入。按键 0 没有连接编码器输入,因为没有其他任何按键被按下时,BCD 输出表示的是 0。

编码器的 BCD 反码输出进入一个存储器,每一个连续的 BCD 代码都会被存储起来,直到全部数字输入完毕。在以后的章节里,还会介绍存储 BCD 数字和二进制数据的方法。

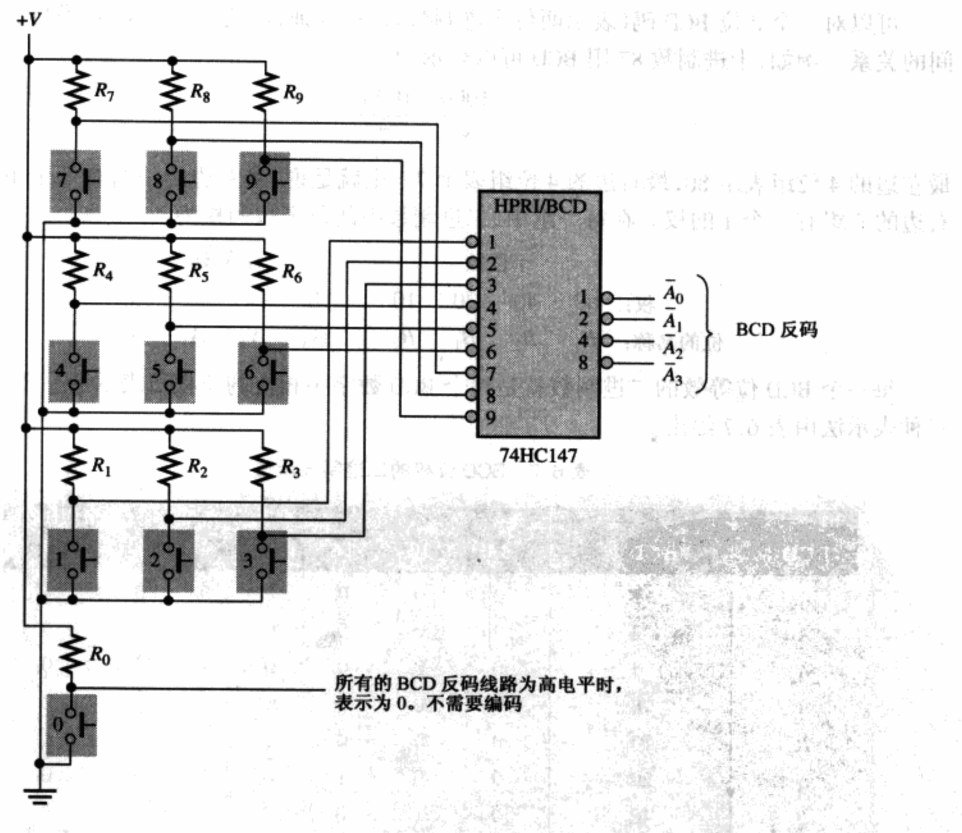


图 6.42　一个简单的键盘编码器

6.7 代码转换器

在本节里,将验证使用逻辑电路组合把一种代码转换成另一种代码的一些方法。
学完本节以后,应当能够

- 解释把 BCD 转换成二进制数的过程
- 使用异或门实现在二进制代码和格雷码之间的相互转换

6.7.1 BCD-二进制转换

BCD-二进制代码转换的一种方法就是使用加法电路。这种基本转换的过程如下:

1. 用一个二进制数字来表示 BCD 数字中每个位的数值或权。
2. 在 BCD 数字中,所有用二进制表示的位的权是 1 的相加。
3. 相加的结果就是 BCD 数字为等效的二进制数。

更简洁的运算描述是

BCD 位的权所表示的二进制数字相加,相加的结果就是完整的二进制数。

可以对一个 8 位 BCD 码(表示两位十进制数)进行验证,以便于理解 BCD 码与二进制数之间的关系。例如,十进制数 87 用 BCD 可以表示为

$$\underbrace{1000}_{8}\ \underbrace{0111}_{7}$$

最左边的 4 位组表示 80,最右边的 4 位组表示 7。也就是说,最左边的一组有一个 10 的权,最右边的一组有一个 1 的权。在每一组中,二进制数中的每个位的权为

	十位数				个位数			
权:	80	40	20	10	8	4	2	1
位的名称:	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0

每一个 BCD 位等效的二进制数就是整个 BCD 数字中相应的位权所表示的一个二进制数。三种表示法由表 6.7 给出。

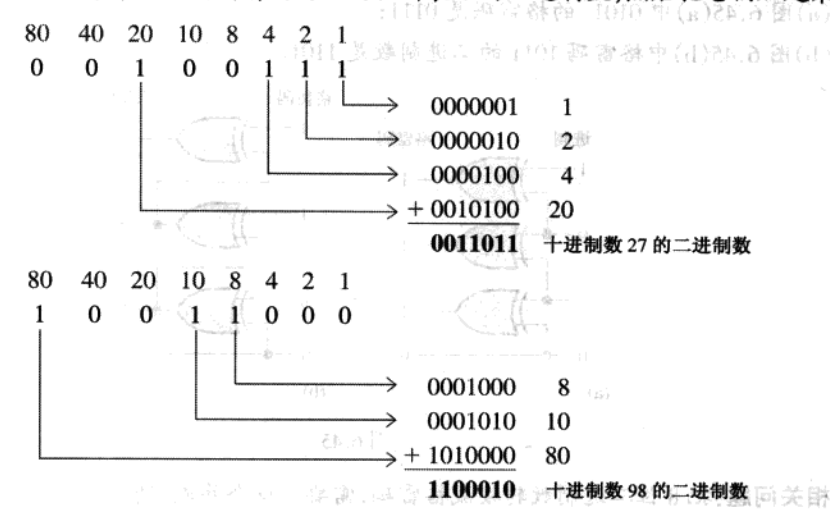
表 6.7 BCD 位权的二进制表示法

BCD 位	BCD 权	64 最高有效位	32	16	8 二进制表示	4	2	1 最低有效位
A_0	1	0	0	0	0	0	0	1
A_1	2	0	0	0	0	0	1	0
A_2	4	0	0	0	0	1	0	0
A_3	8	0	0	0	1	0	0	0
B_0	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
B_2	40	0	1	0	1	0	0	0
B_3	80	1	0	1	0	0	0	0

如果在这个 BCD 数字中，把所有为 1 的权所表示的二进制数相加，其结果就是与此 BCD 数相对应的二进制数。例 6.12 对此给出解释。

例 6.12 把 BCD 数字 00100111(十进制数为 27)和 10011000(十进制数为 98)转换为二进制数。

解：写出在这个数字中出现的所有的为 1 的权所表示的二进制数，然后把它们加起来。



相关问题：写出将 BCD 数字 01000001 转换为二进制数的过程。

记住这个基本过程后，来看看如何使用逻辑电路实现这个过程。一旦 BCD 数字中每个 1 的二进制表达式确定下来后，就可以使用加法电路把表达式中的每一列 1 相加。只有当相对应的 BCD 位为 1 时，每列才会出现 1。因此，BCD 中出现的 1 用于产生相应二进制数的 1，这个 1 处在加法运算的相应列上。为了处理一个两位十进制数(两个一位十进制数)的 BCD 代码，需要 8 条 BCD 输入线和 7 条二进制输出线(表示 99 以内的二进制数需要 7 位)。

6.7.2 二进制 – 格雷码和格雷码 – 二进制转换

在第 2 章中，已经介绍过格雷码 – 二进制转换的基本过程。这种转换需要使用异或门。其中的代码转换还可以用可编程逻辑器件编程实现。图 6.43 为一个 4 位二进制 – 格雷码转换器，图 6.44 为一个 4 位格雷码 – 二进制转换器。

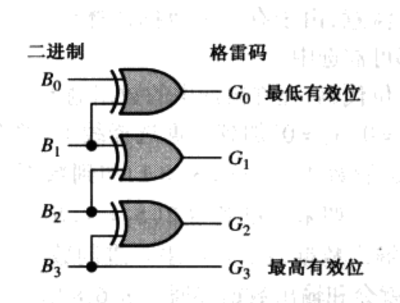


图 6.43 4 位二进制 – 格雷码转换逻辑

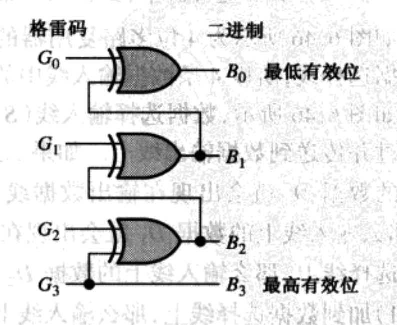


图 6.44 4 位格雷码 – 二进制转换逻辑

例 6.13

(a)使用异或门把二进制数字 0101 转换成格雷码;

(b)使用异或门把格雷码 1011 转换成二进制数字。

解:

(a)图 6.45(a)中 0101_2 的格雷码是 0111;

(b)图 6.45(b)中格雷码 1011 的二进制数是 1101_2。

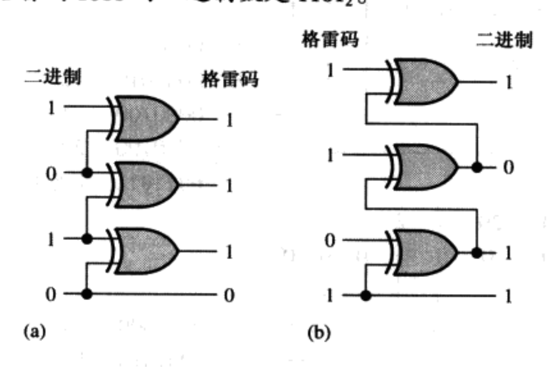


图 6.45

相关问题:把 8 位二进制数转换成格雷码,需要多少个异或门?

6.8 多路复用器(数据选择器)

多路复用器芯片允许数字信号从几个数据源通过一条单一线路传送到一个共同的地方。基本的多路复用器有几条数据输入线和一条单一的输出线。它还有数据选择输入线,用于把输入线中任何一条线上的数据和输出线连通。所以,多路复用器也称为数据选择器。

学完本节以后,应当能够

- 解释多路复用器的基本操作过程
- 描述 74LS151 和 74HC157 这两种多路复用器
- 扩展多路复用器以处理更多的数据输入
- 把多路复用器用做一个逻辑函数发生器

◇ 在一个多路复用器中,数据从几条线路传送到一条线路上

如图 6.46 所示为 4 位多路复用器的逻辑符号。注意,由于有两个数据选择位,所以有两条数据选择线,所以 4 条数据输入线中的任何一条都可被选中。

如图 6.46 所示,数据选择输入线(S)上的一个 2 位代码,允许被选中的数据输入线上的数据通过并传送到数据输出线上。如果二进制数 0($S_1=0, S_0=0$)加到数据选择线上,那么输入线上的数据 D_0 就会出现在输出数据线上。如果二进制数 1($S_1=0, S_0=1$)加到数据选择线上,那么输入线上的数据 D_1 就会出现在输出数据线上。如果二进制数 2($S_1=1, S_0=0$)加到数据选择线上,那么输入线上的数据 D_2 就会出现在输出数据线上。如果二进制数 3($S_1=1$, $S_0=1$)加到数据选择线上,那么输入线上的数据 D_3 就会和输出数据接通。表 6.8 给出了这种运算的一个总结。

现在,观察一下完成这种多路复用操作所需的逻辑电路。数据输出与被选中的数据输入是相同的。因此,可以得到一个关于数据输入和选择输入的输出逻辑表达式。

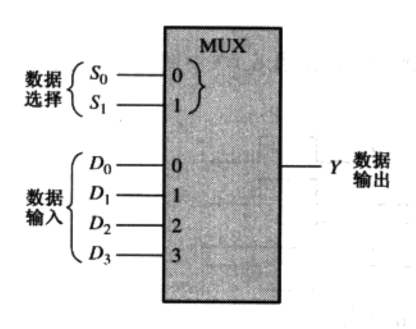


图 6.46　4 选 1 数据选择器/多路复用器的逻辑符号

表 6.8　4 选 1 多路复用器的数据选择

数据选择输入		选中的输入
S_1	S_0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

仅当 $S_1=0$ 和 $S_0=0$ 时,数据输出和 D_0 相等:$Y=D_0\overline{S_1}\,\overline{S_0}$。
仅当 $S_1=0$ 和 $S_0=1$ 时,数据输出和 D_1 相等:$Y=D_1\overline{S_1}S_0$。
仅当 $S_1=1$ 和 $S_0=0$ 时,数据输出和 D_2 相等:$Y=D_2S_1\overline{S_0}$。
仅当 $S_1=1$ 和 $S_0=1$ 时,数据输出和 D_3 相等:$Y=D_3S_1S_0$。

把这些项相或,数据输入的总的表达式为

$$Y = D_0\bar{S}_1\bar{S}_0 + D_1\bar{S}_1S_0 + D_2S_1\bar{S}_0 + D_3S_1S_0$$

这个等式的实现需要 4 个三输入与门、1 个四输入或门和两个反相器,两个反相器用来产生 S_1 和 S_0 的反码,如图 6.47 所示。由于数据可以从输入线路中的任一条进行选择,所以也把这种电路看成是一个数据选择器(data selector)。

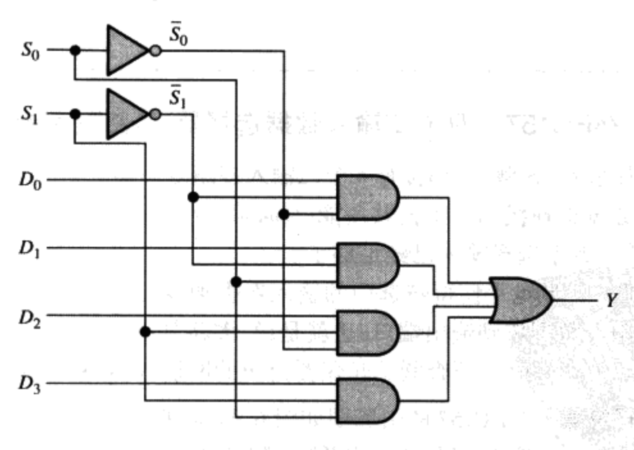


图 6.47　四输入多路复用器的逻辑图



计算机小知识

总线是计算机的一条内部通路,电子信号从计算机的一端传送到另一端。在计算机网络中,一条共享总

线就是连接系统中所有的计算机、用于交换数据的通路。一条共享总线可以连接存储器和输入/输出设备,这样这些设备就可由系统中所有的计算机共享。对共享总线的访问是由总线判优器(类似于多路复用器)控制的,它每次只允许一台计算机享用系统的总线。

例 6.14 如图 6.48(a)所示的数据输入和数据选择波形,加在图 6.47 中的多路复用器上。请根据数据输入波形来确定输出波形。

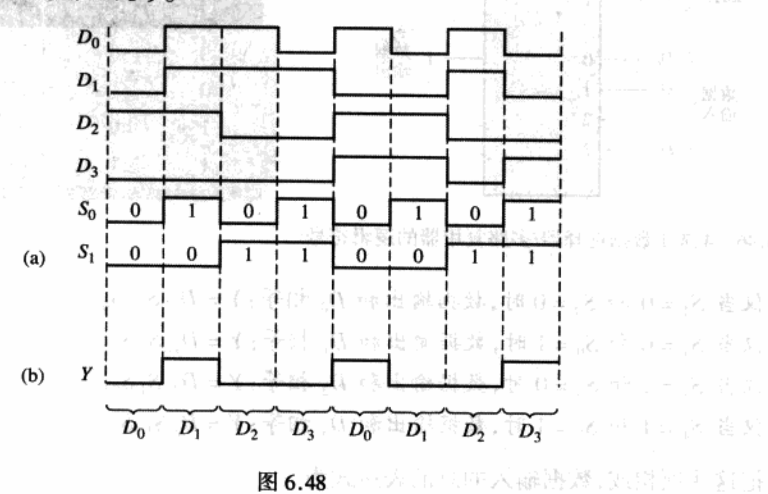


图 6.48

解:在每一个时间间隔期间,数据选择输入端的二进制状态,确定了哪一条数据输入被选中。注意,数据选择输入的二进制状态变化重复二进制序列 00,01,10,11,00,01,10,11,以此类推。所求的输出波形如图 6.48(b)所示。

相关问题:如果将图 6.48 中 S_0 和 S_1 的波形互换,请构建一个时序图,用来表示所有的输入和输出波形。

74HC157 四个二输入数据选择器/多路复用器

74HC157 芯片及它的 LS 型号,都含有 4 个二输入多路复用器。这 4 个多路复用器的每一个都共享一个公共数据选择线和一个公共使能(Enable)端。由于每一个多路复用器只有两个输入被选择,所以单个数据选择输入已经足够了。

$\overline{Enable}$输入端的一个低电平使得被选中的输入数据通过芯片传到输出端。而$\overline{Enable}$输入端的一个高电平则阻止数据传到输出端口;也就是说,多路复用器被禁止工作。这种芯片在其他的 CMOS 或 TTL 系列中也有。请查阅德州仪器公司的网址:www.ti.com。

ANSI/IEEE 逻辑符号 74HC157 的引脚图如图 6.49(a)所示。74HC157 的 ANSI/IEEE 逻辑符号如图 6.49(b)所示。注意,这里的 4 个多路复用器被轮廓分割线分开,这 4 个多路复用器的公共输入由顶上有凹痕的框处标出,称之为公共控制框。在 MUX(多路复用器)框上部的标注都适用于下面其他的框。

注意在 MUX 框中的标注 1 和 $\overline{1}$ 及公共控制框中的 G_1。这些标注是 ANSI/IEEE 标准 91-1984确定的关联标注系统的一个例子。在这种情况下,G_1 是表示数据选择输入和标注为 1

或$\bar{1}$的数据输入的与运算关系。($\bar{1}$的意思是与的关系应用于 G_1 输入的反码。)换句话说,当数据选择输入端为高电平时,多路复用器的 B 输入线被选中;当数据选择输入端为低电平时,多路复用器的 A 输入线被选中。"G"永远用来表示与关联。在本书的适当地方,还会介绍关联表示的其他方面。

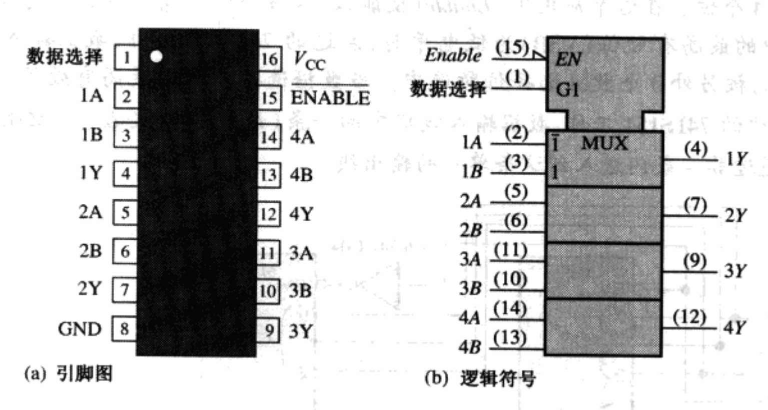


图 6.49　74HC157 四 - 二输入数据选择器/多路复用器的引脚图和逻辑符号

74LS151 八输入数据选择器/多路复用器

74LS151 有 8 个数据输入($D_0 \sim D_7$),因此有 3 条数据选择或地址输入线($S_0 \sim S_2$)。为了选择 8 个数据输入($2^3 = 8$)中的任何一个,需要三个位。$\overline{Enable}$输入的低电平使得被选中的数据输入传送到输出端。注意,数据输出和它的反码都可得到。引脚图如图 6.50(a)所示,ANSI/IEEE 的逻辑符号如图 6.50(b)所示。在这种情况下,逻辑符号中不需要一个公共控制框,因为受控制的仅仅只有一个多路复用器,而不像 74HC157 那样有 4 个。逻辑符号中的标注表示的是数据选择输入和从 0 到 7 的每一个数据输入相与的运算关系。这种芯片在其他的 CMOS 或 TTL 系列中也有。请查阅德州仪器公司的网址:www.ti.com。

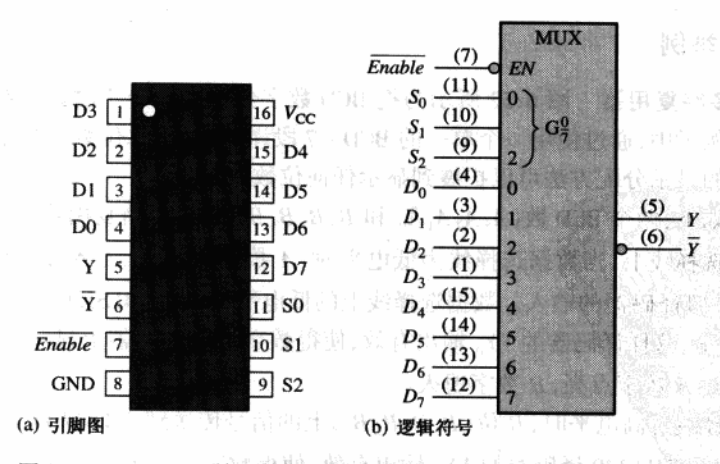


图 6.50　74LS151 八输入数据选择器/多路复用器的引脚图和逻辑符号

例 6.15 使用 74LS151 及其他任何需要的逻辑,把 16 条数据线路分配到一条单一的数据输出线上。

解:实现这个系统的方法如图 6.51 所示。为了选择 16 条数据输入线($2^4=16$)的其中一条,需要 4 个位。在这个应用中,$\overline{Enable}$(使能)输入被用做最高有效数据选择位。当数据选择码中的最高有效位(MSB)为低电平时,左边的 74LS151 工作,数据输入线中的一条($D_0\sim D_7$)被另外 3 个数据选择位所选中。当数据选择码中的最高有效位(MSB)为高电平时,右边的 74LS151 工作,数据输入线路中的一条($D_8\sim D_{15}$)被选中。这时被选中的输入数据通过非-或门进入到这条单一的输出线。

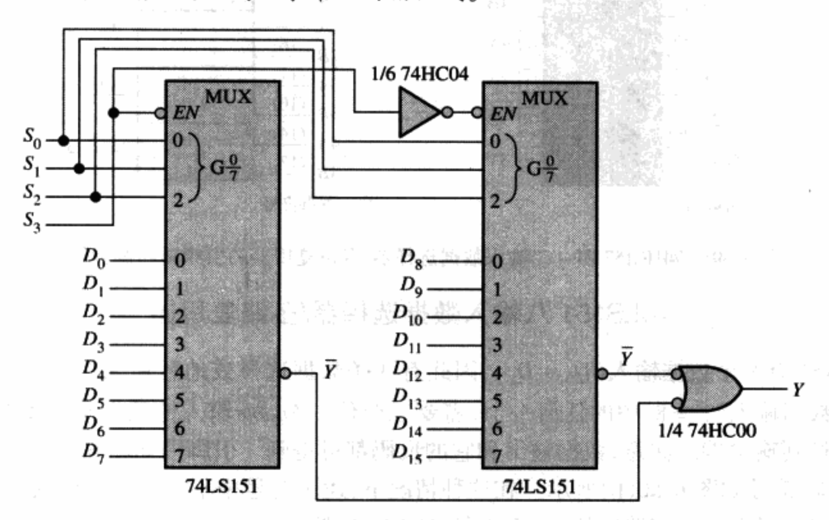


图 6.51 一个 16 输入多路复用器

相关问题:当需要选择数据输入 D_0、D_4、D_8 和 D_{13} 时,请确定选择输入的代码。

6.8.1 应用举例

7 段显示多路复用器 图 6.52 所示为把 BCD 数字分配到一个 7 段显示器上的一个简单方法。在这个例子中,通过使用一个单一的 BCD-7 段译码器,把二位数字显示在 7 段显示器上。这个基本的显示分配方法可以扩展到显示任何位数。

基本过程如下:两个 BCD 数($A_3A_2A_1A_0$ 和 $B_3B_2B_1B_0$)加到多路复用器的输入。一个方波信号加在数据选择线上,当数据选择线为低电平时,A 数字的位($A_3A_2A_1A_0$)上的信号传送到 74LS47 BCD-7 段译码器的输入。数据选择线上的低电平也使得 74LS139 2 线-4 线译码器的 A_1 输入为低电平,这样译码器的 $1Y_0$ 输出有效,使得数字 A 在显示器上显示,显示器实际上是共阴连接。这时 A 数字点亮,B 数字熄灭。

当数据选择线为高电平时,B 位($B_3B_2B_1B_0$)上的信号传送到 74LS47 BCD-7 段译码器的输入。同样,由于 74LS139 译码器的 $1Y_1$ 输出有效,使得数字 B 在显示器上显示,这时 B 数字点亮,A 数字熄灭。重复的周期为数据选择端方波信号的频率。这个频率必须足够高(大约 30 Hz),以避免因数据的分配而产生视觉上的闪烁。

逻辑函数发生器　数据选择器/多路复用器的一种有效应用就是乘积之和形式的组合逻辑函数发生器。当以这种方式使用时，它可以代替一些分立的逻辑门，常常可以大大地减少集成电路的数量，并且可以使电路设计变得更简单。

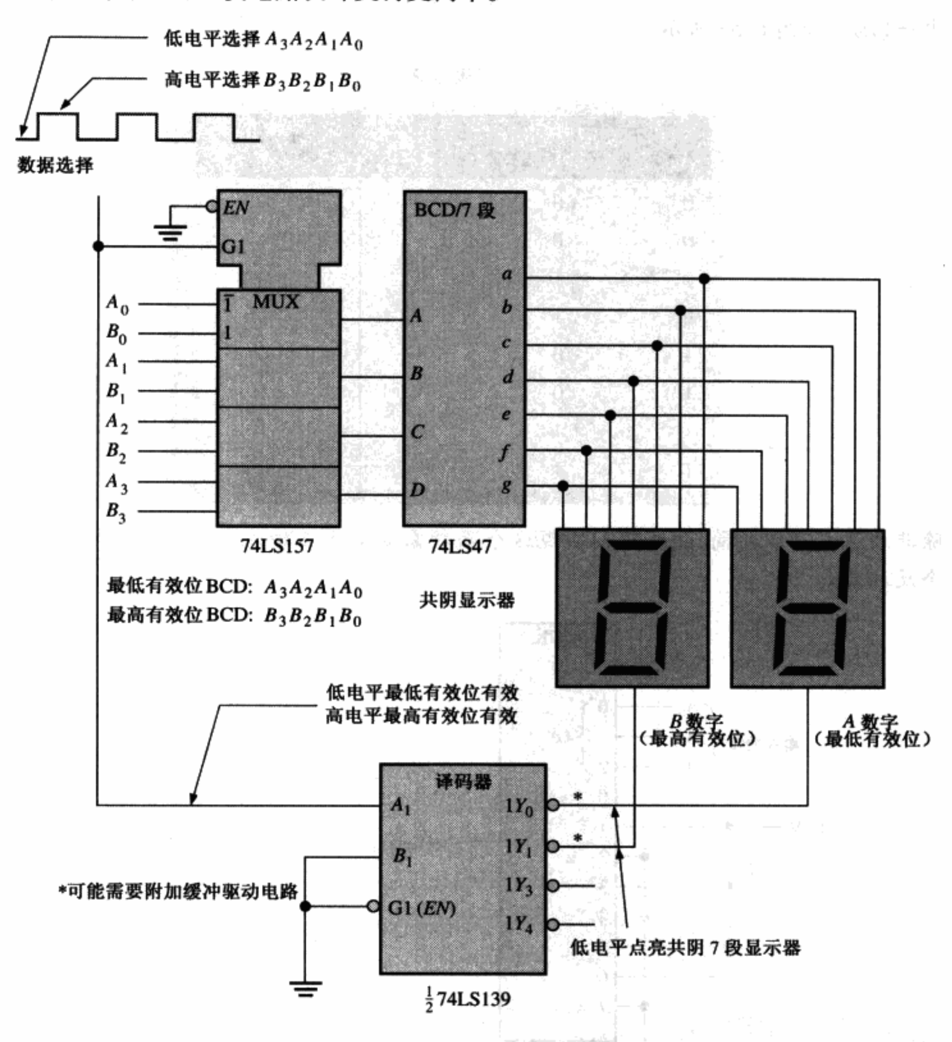


图 6.52　简单的 7 段显示器多路复用逻辑

为了进行解释，可以用一个 74LS151 八输入数据选择器/多路复用器来实现任何具体的 3 变量逻辑函数，这时三个变量连接到数据选择输入端，每一个数据输入端连接所需的电平，这些电平由真值表中对应的输出函数所确定。例如，当三个变量的组合是 $\overline{A_2}A_1\overline{A_0}$、输出函数为 1 时，数据输入端 2（由 010 选中）连接到高电平。当这个特定的变量组合出现在数据选择输入线上时，数据输入端 2 上的高电平传送到输出。这个例子将对阐明这个应用有所帮助。

例 6.16　如表 6.9 所示，使用一个 74LS151 八输入数据选择器/多路复用器来实现具体的逻辑函数。并和用分立逻辑门实现的方法进行比较。

解:注意,从这个真值表可以看出,如下的输入变量组合:001、011、101 和 110 对应输出 Y 为 1。而对于其他的组合,Y 为 0。因为这个函数由数据选择器来实现,所以上述的每一个组合选中的数据输入端必须连接到高电平(5 V)。所有其他的数据输入端则需连接低电平(地),如图 6.53 所示。

表 6.9

输入			输出
A_2	A_1	A_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

除非表达式可以化简,用逻辑门实现这个函数需要 4 个三输入与门、1 个四输入或门和 3 个反相器。

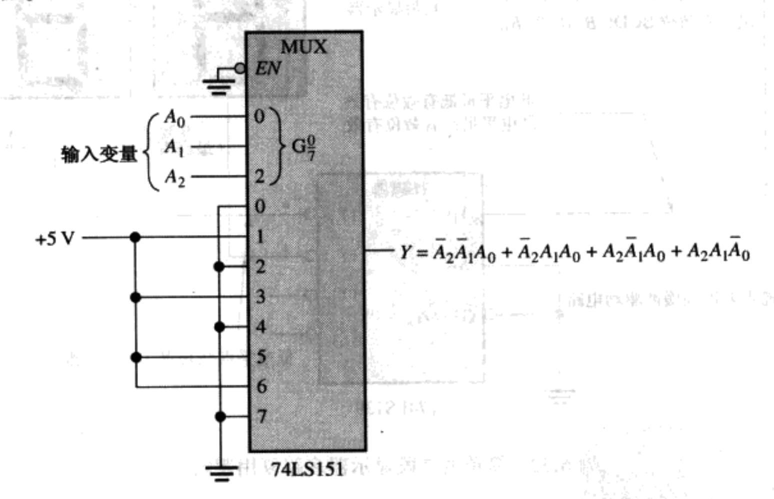


图 6.53 数据选择器/多路复用器连接组成一个 3 变量逻辑函数发生器

相关问题:使用 74LS151 实现下列表达式:

$$Y = \bar{A}_2\bar{A}_1\bar{A}_0 + A_2\bar{A}_1\bar{A}_0 + \bar{A}_2A_1\bar{A}_0$$

例 6.16 解释了如何把一个八输入数据选择用做 3 变量逻辑函数发生器。实际上,使用连接数据输入的一个位(A_0),还可以将这种芯片用做一个 4 变量逻辑函数发生器。

一个 4 变量真值表有 16 个输入变量组合。当使用一个 8 位数据选择器时,每一个输入都

会被选择两次：第一次 A_0 为0，第二次 A_0 为1。记下这个结论，可以运用下列规则（Y 为输出，A_0 为最低有效位）：

1. 如果某个输入变量的组合 $A_3A_2A_1$ 在 $Y=0$ 时两次都选中了一个给定的数据输入端，那么把此数据输入端连接到地(0)；
2. 如果某个输入变量的组合 $A_3A_2A_1$ 在 $Y=1$ 时两次都选中了一个给定的数据输入端，那么把此数据输入端连接到电压 $V(1)$；
3. 如果某个输入变量的组合 $A_3A_2A_1$ 在 $Y=0$ 和 $Y=1$ 时两次都选中了一个给定的数据输入端，并且 $Y=A_0$，那么把此数据输入端连接到 A_0；
4. 如果某个输入变量的组合 $A_3A_2A_1$ 在 $Y=0$ 和 $Y=1$ 时两次都选中了一个给定的数据输入端，并且 $Y=\overline{A_0}$，那么把此数据输入端连接到$\overline{A_0}$。

例6.17 使用一个74LS151八输入数据选择器/多路复用器实现表6.10中的逻辑函数。并和用分立逻辑门实现的方法进行比较。

表6.10

十进制数	输入				输出
	A_3	A_2	A_1	A_0	Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

解：数据选择输入是 $A_3A_2A_1$。在表6.10的第1行中，$A_3A_2A_1=000$，$Y=A_0$。在第2行中，还是 $A_3A_2A_1=000$，$Y=A_0$。因此，A_0 与输入端0相连。在第3行中，$A_3A_2A_1=001$，此时 $Y=\overline{A_0}$。同样，在第4行中，$A_3A_2A_1=001$，$Y=\overline{A_0}$。因此，A_0 被反相并与输入端1相连。根据具体的规律，这样的发现会一直持续下去直到与每个输入得到恰当的连接，如图6.54所示。

如果使用逻辑门实现,函数需要多达 10 个四输入与门、一个十输入或门和 4 个反相器。当然,可能的简化可以减少一些逻辑门。

相关问题:在表 6.10 中,如果输入都为 0 时,$Y=0$,并将表格剩余行中 Y 的 1 和 0 交替排列,使用 74LS151 来实现这个逻辑函数。

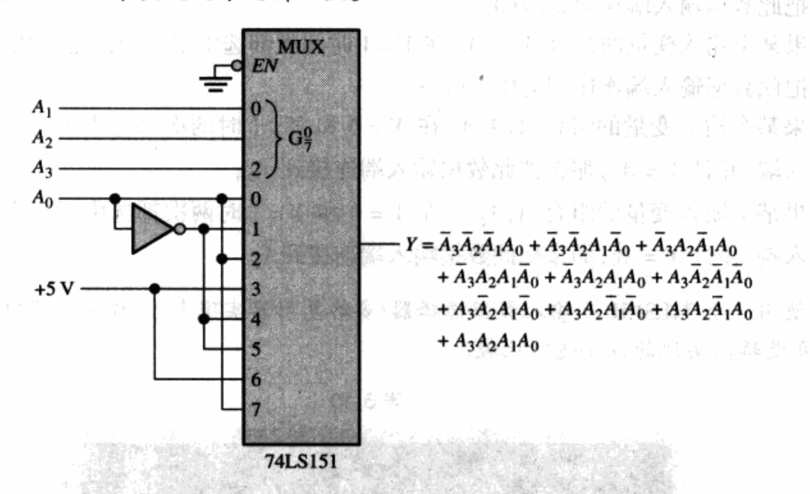


图 6.54 数据选择器/多路复用器相连组成一个 4 变量逻辑函数发生器

6.9 多路分配器

多路分配器(DEMUX)的逻辑功能与多路复用器的逻辑功能基本上相反。它把一条线路上的数字信息分配到一定数量的输出线上。由于这个原因,多路分配器也称为数据分配器。本节将了解到译码器也可以用做多路分配器。

学完本节以后,应当能够

- 解释一个多路分配器的基本原理
- 描述如何把 74HC154 4 线 - 16 线译码器用做一个多路分配器
- 使用具体数据和数据选择输入绘制多路分配器的时序图

◇ 在一个多路分配器中,数据从一条线路中被分配到几条线路上。

1 线 - 4 线多路分配器(DEMUX)的电路如图 6.55 所示。数据输入线与所有的与门连接。每次两条数据选择线只开通一个逻辑门,然后数据输入线上的数据通过这个选中的门传送到相应的数据输出线上。

例 6.18 串行数据输入波形(数据输入)和数据选择输入(S_0 和 S_1)如图 6.56 所示。试根据图 6.55 所示的这个多路分配器,确定从 D_0 到 D_3 的数据输出波形。

解:输出波形如图 6.56 所示。注意,数据选择线通过一个二进制序列,使得每一个连续的输入位按照 D_0、D_1、D_2 和 D_3 的顺序变化。

相关问题:如果时序图 S_0 和 S_1 波形反相,绘制这个多路复用器的时序图。

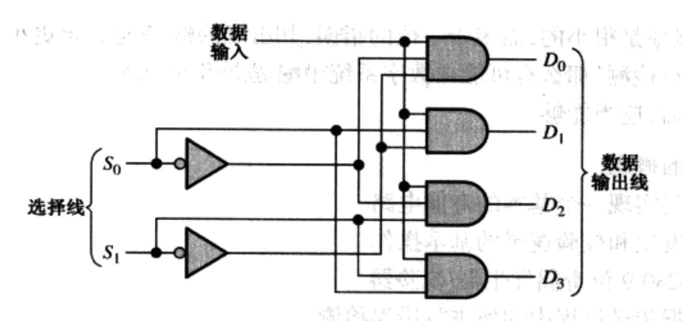


图 6.55 一个1线-4线多路分配器

74HC154 多路分配器

前面介绍了 74HC154 译码器作为一个 4 线 - 16 线译码器的应用(参见 6.5 节)。这种器件及其他的译码器也可以用做多路分配器。当用做多路分配器时的逻辑符号如图 6.57 所示。在多路分配器的应用中,输入线用做数据选择线。其中一个片选输入用做数据输入线,其他片选输入为低电平时,使得图底部的内部非 - 与门工作。这种芯片在其他的 CMOS 或 TTL 系列中也会出现。请查阅德州仪器公司的网址:www.ti.com。

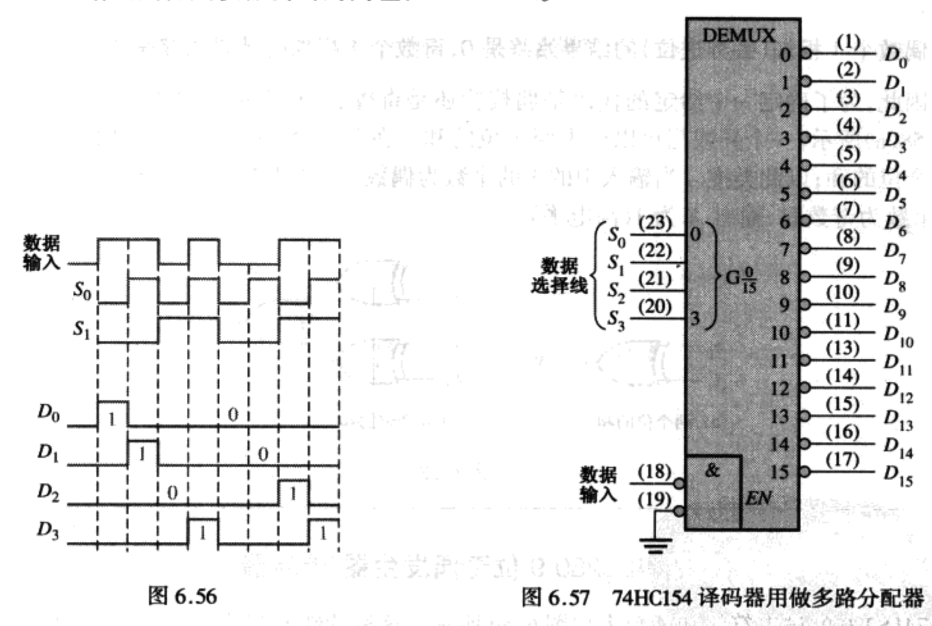


图 6.56

图 6.57 74HC154 译码器用做多路分配器

6.10 奇偶发生器/校验器

当数据码在一个系统中从一个点传送到另一个点,或者这些数据码由一个系统传送到另一个系统时,错误就可能发生。这些错误就是组成代码信息的位发生不需要的变化;也就是说,由于器件的故障或电子噪声,1 会变成 0,或由 0 变成 1。在多数数字系统中,甚至只有一位

的错误,出现的概率是很小的,而多于一位的错误,其出现的概率是甚至更小。尽管如此,如果我们不对错误进行检测,那么有可能在数字系统中酿成严重的问题。

学完本节以后,应当能够

- 解释奇偶的概念
- 使用异或门实现一个基本的奇偶电路
- 描述奇偶发生和校验逻辑的基本操作
- 描述 74LS280 9 位奇偶发生器/校验器
- 讨论在数据传送过程中如何进行错误检测

在第 2 章中已经讲过,把一个奇偶校验位放于一组信号位中,使所有的 1 加起来为偶数或奇数(根据系统的情况而定),就是错误检测的奇偶检验方法。除了校验位,几种特殊的码还提供固有误差的检测。

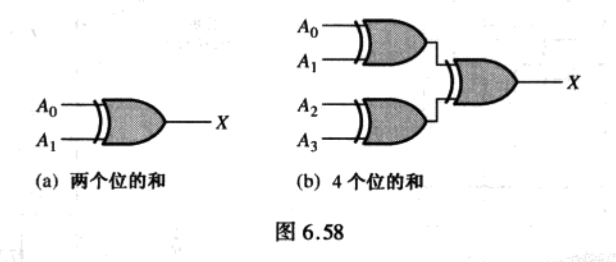
6.10.1 基本奇偶逻辑

◇ 用做错误检测的奇偶校验位指出在一个代码中,1 的个数是偶数还是奇数。

为了对一个给定的代码进行检测或者产生合适的奇偶校验,可以使用下列给出的基本原理:

偶数个 1 相加(丢弃进位)的结果始终是 0,奇数个 1 相加的结果始终是 1。

因此,为了确定一个给定的代码是偶校验还是奇校验,只需把代码中所有的位相加。如图 6.58(a)所示,一个异或门可以产生两个位的和。在图 6.58(b)中,3 个异或门相连,可以产生 4 个位的和;以此类推。当输入中的 1 的个数为偶数时,输出 X 为 0(低电平)。当输入中的 1 的个数为奇数时,输出 X 为 1(高电平)。



(a) 两个位的和　　(b) 4 个位的和

图 6.58

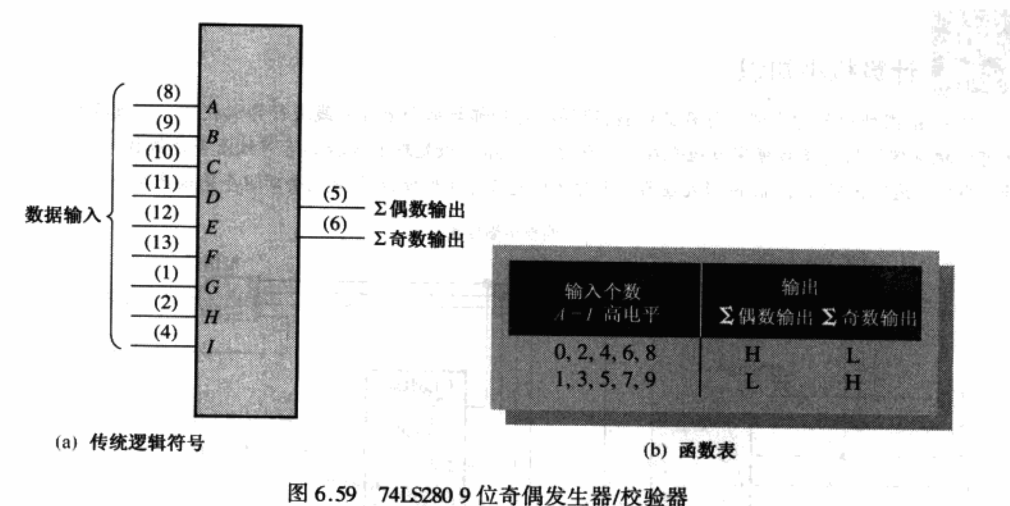
74LS280 9 位奇偶发生器/校验器

74LS280 的逻辑符号和函数表如图 6.59 所示。这种特殊的芯片可以对一个 9 位代码(8 个数据位和一个校验位)进行奇校验或偶校验,或者还可以对一个 9 位二进制代码产生一个校验位。从 A 到 I 为输入;当输入端有偶数个 1 时,Σ偶数输出为高电平,Σ奇数输出为低电平。这种芯片在其他的 CMOS 或 TTL 系列中也会出现。请查阅德州仪器公司的网址:www.ti.com 。

奇偶校验器　当把这种芯片用做一个偶校验器时,输入位 1 的个数应该始终为偶数;当出现奇偶错误时,Σ偶数输出为低电平,Σ奇数输出为高电平。当把这种芯片用做一个奇校验器

时，输入位1的个数应该始终为奇数；当出现奇偶错误时，Σ奇数输出为低电平，Σ偶数输出为高电平。

奇偶发生器 如果把这种芯片用做一个偶发生器，那么奇偶校验位在Σ奇数输出获得。因为如果输入位有偶数个1，则这个输出为0；反之，如果输入位1的个数为奇数，则这个输出为1。当把这个器件用做一个奇发生器时，奇偶校验位在Σ偶数输出获取，因为当输入位1的个数为奇数时，则Σ偶数输出为0。



输入个数 A-I 高电平	输出 Σ偶数输出	Σ奇数输出
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H

图6.59 74LS280 9位奇偶发生器/校验器

6.10.2 数据传输系统的错误检测

一个简单的数据传输系统如图6.60所示，用于阐明奇偶发生器/校验器的应用，以及多路复用器和多路分配的应用，还用它来解释某些应用中所需要的数据存储。

在此应用中，七个数据源中的数字数据被多路复用器合并到一条单一的线上，用做远距离传输。七个数据位(从 D_0 到 D_6)加到多路复用器的数据输入端，与此同时还加到偶发生器的输入端。奇偶发生器的Σ奇数输出用做偶校验位。如果输入 A 到 I 的1的个数为偶数，则这个偶校验位为0；如果输入 A 到 I 的1的个数为奇数，则这个偶校验位为1，这个位就是被传输码的 D_7。

数据选择输入在一个二进制序列里循环重复，从 D_0 开始，每一个位以串行方式传输到传输线($\bar{Y}$)。在这个例子中，传输线路由4根导线组成：1根携带串行数据，3根携带时序信号(数据选择)。当然，还有一些更复杂的传送时序信号的方法，不过为了解释基本原理，我们还是使用这个直接的方法。

在系统结尾部分的多路分配器中，将数据选择信号和串行数据流加在这个多路分配器上。多路分配器把数据位分配到输出线上，以便这些数据出现在多路分配器的输出端。也就是说，D_0 出现在 D_0 输出上；D_1 出现在 D_1 输出上，以此类推。奇偶校验位出现在 D_7 输出上。这8个位暂时存储下来，并加到偶校验器上。当奇偶校验位 D_7 出现并被存储时，奇偶校验器的输入端才会出现所有的位。在这同时，数据选择码111打开与门(检错门)。如果奇偶性正确，

∑偶数输出 0，检错门的输出(出错输出端)仍然为 0。如果奇偶性是错误的，检错门输入端全部是 1，结果检错门的输出(出错输出端)为 1。

这种特殊的应用演示了数据存储的必要性，使我们更加深入地了能存储器件的用途。这些概念将在第 7 章介绍并在后面的章节中使用。

图 6.61 的时序图描述了特定情况下传输两个 8 位字，其中一个正确接收，另一个有一位错误。



计算机小知识

Pentium 微处理器对内部进行奇偶校验，同样也对外部数据和地址总线进行奇偶校验。在读操作时，外部系统可以把奇偶性信息及数据字节组合在一起传送。Pentium 微处理器校验其奇偶性是否是偶数，并输出相应的信号。当输出地址代码时，Pentium 微处理器不会对地址进行奇偶性校验，但是它会对这个地址产生一个偶校验位。

四根传输导线　检错门　出错 = 1

MUX　EN　$G\frac{0}{7}$　74LS151　偶校验位　偶校验发生器 (74LS280)　Σ奇

DEMUX　$G\frac{0}{7}$　74LS138　偶校验位(D_7)　存储器*　奇校验发生器 (74LS280)　Σ偶

*存储器在第9章介绍，并在后面的章节中用到

图 6.60　简单数据传输系统的错误检测

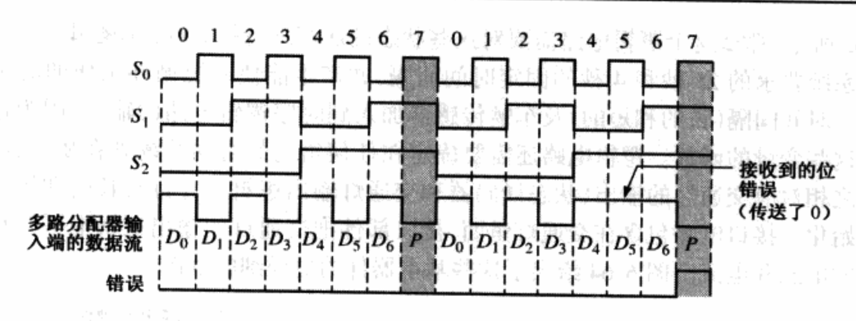


图 6.61 图 6.60 中系统有错误和无错误的数据传输示例

6.11 数字系统应用

在数字系统应用中，会接触到一个交通灯控制系统。在本节中，这个系统需要的条件已经确立。设计一个总体框图，并创建一个状态图，用于确定操作的顺序。这个系统的一部分涉及到设计组合逻辑电路及检测方法的考虑。系统的定时和时序部分将在第 7 章和第 8 章中介绍。

6.11.1 总体系统需要的条件

为了对一条繁忙主干道和一条较为冷落的边道的十字路口处的交通灯进行控制，需要一个数字控制器。主干道上的绿灯至少亮 25 秒，或者直到边道上有车辆为止。边道上的绿灯持续亮到没有车辆为止，或绿灯最多亮 25 秒。主干道和边道在从绿灯亮变化到红灯亮的时间间隔中，有 4 秒时间的警示灯亮（黄灯）。这些必要条件如图 6.62 所示。

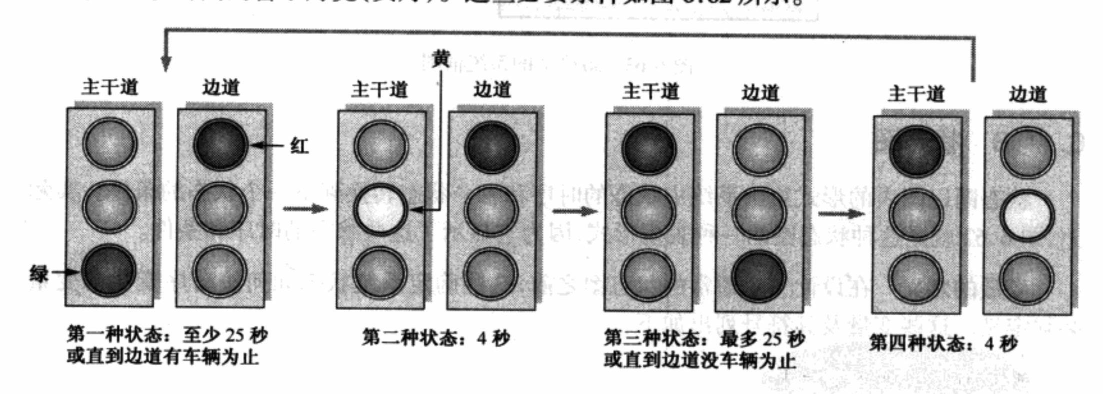


图 6.62 交通灯时序图所需要的条件

6.11.2 设计系统框图

根据这些条件，可以设计这个系统的一个框图。首先，了解到这个系统必须控制六对不同的灯。主干道的两个方向有一对红灯、一对绿灯和一对黄灯；同样，边道的两个方向也有一对红灯、一对绿灯和一对黄灯。另外，还有一个来自边道车辆传感器的外部输入（除电源外），图 6.63是一个满足这些条件的最简洁的框图。

使用这个最简洁的系统框图，可以开始进行一些更为细致的工作了。这个系统有 4 种状

态,如图 6.62 所示,那么这个逻辑电路需要对这些状态的次序进行控制(时序逻辑)。另外,电路需要产生系统要求的 25 秒和 4 秒的固定时间间隔,并产生能使系统循环工作的时钟信号(定时电路)。时间间隔(长的和短的)及车辆传感器加入到时序逻辑电路的输入,因为这些状态的时序是这些变量的函数。逻辑电路还需要确定在任何给定的时间系统处在哪一种状态,以便产生与之相对应交通灯的输出(状态译码器和交通灯输出逻辑),并且对长的和短的时间间隔进行初始化。接口电路包含在交通灯里面,接口部件把交通灯的输出逻辑电平转换为每个灯所需要的电压和电流。图 6.64 给出了这些基本器件的更详细的框图。

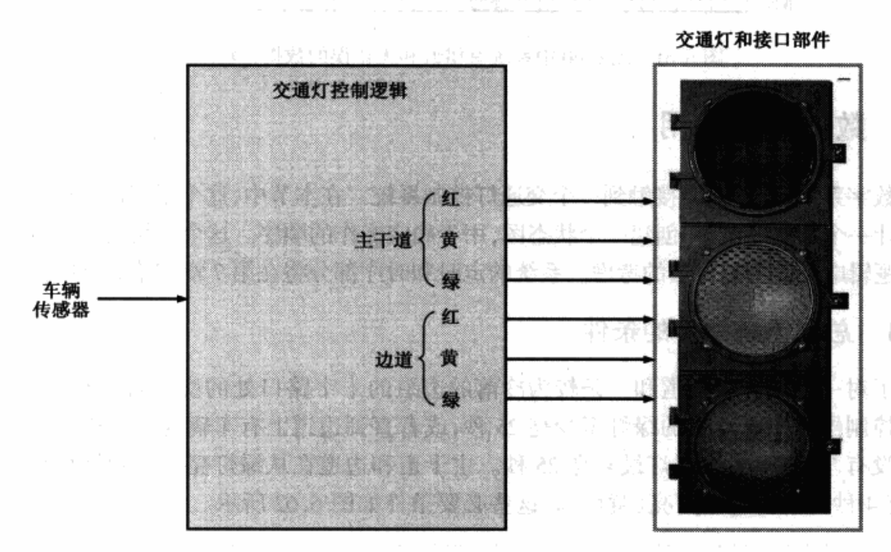


图 6.63 最简洁的系统框图

6.11.3 状态图

状态图以图表的形式给出系统中状态的时序和一个状态转换到下一个状态的条件。实际上,图 6.62 就是这种状态图的一种表现形式,因为它显示了这些状态的时序和条件。

变量的定义 在设计一个通常的状态图之前,这些确定系统状态如何按时序变化的变量必须定义。这些变量及其符号列出如下:

- 边道上出现车辆 = V_s
- 25 秒(长定时器) 启动 = T_L
- 4 秒(短定时器) 启动 = T_S

使用变量的反码表示相反的条件。例如,$\overline{V}_s$ 表示边道上没有车辆,$\overline{T}_L$ 表示长定时器关闭,$\overline{T}_S$ 短定时器关闭。

状态图的描述 状态图如图 6.65 所示。根据圆圈中的 2 位格雷码时序来标注 4 种状态。每种状态的循环箭头表示在相关的变量或表达式确定的情况下,系统的状态保存不变。每个从一种状态到下一种状态的箭头表示在相关的变量或表达式确定的情况下的状态变化。

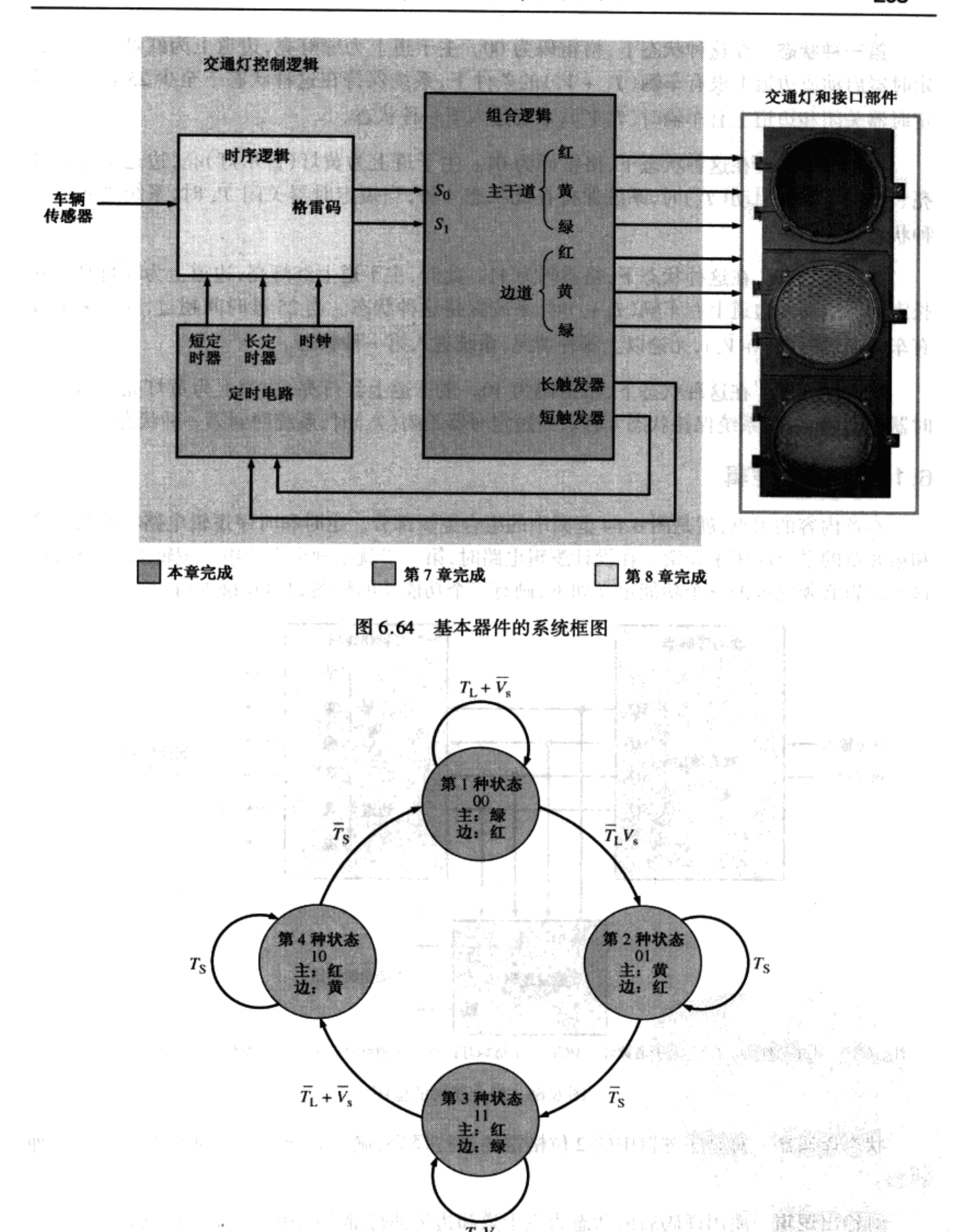


图 6.64　基本器件的系统框图

图 6.65　给出格雷码顺序的交通灯控制系统的状态图

第一种状态 在这种状态下,格雷码为00。主干道上为绿灯亮,边道上为红灯亮。在长定时器启动或边道上没有车辆($T_L + \bar{V}_s$)的条件下,系统保持在这种状态下至少25秒。当长定时器关闭和边道上有车辆时($\bar{T}_L V_s$),系统进入下一种状态。

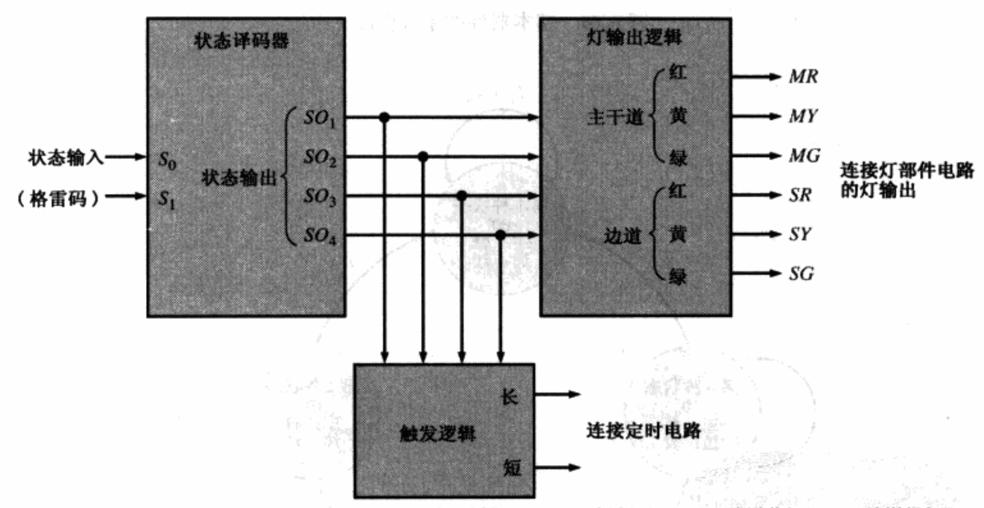
第二种状态 在这种状态下,格雷码为01。主干道上为黄灯(警示灯)亮,边道上为红灯亮。当短定时器启动(T_S)时,系统保持在此状态4秒,当短定时器关闭 $\bar{T}_S$ 时,系统进入下一种状态。

第三种状态 在这种状态下,格雷码为11。此时,主干道上红灯亮,边道上为绿灯亮。在长定时器启动和边道上有车辆($T_L V_s$)时,系统保持这种状态。当25秒时间超过,或边道上没有车辆通过时($\bar{T}_L + \bar{V}_s$),无论以上哪种情况,系统进入第一种状态。

第四种状态 在这种状态下,格雷码为10。主干道上红灯亮,边道上为黄灯亮。当短定时器启动(T_S)时,系统保持状态4秒。当短定时器关闭($\bar{T}_S$)时,系统回到第一种状态。

6.11.4 组合逻辑

本章内容的重点,就是图6.64框图中的组合逻辑部分。定时和时序逻辑电路将在第7章和第8章的系统应用中介绍。在设计逻辑电路时,第一步就是画出系统组合逻辑部分的框图。这个逻辑必须完成的三个功能定义如下,画有三个功能块的框图如图6.66所示。



注:(*MR*:主干道红灯,*MY*:主干道黄灯,*MG*:主干道绿灯;*SR*:边道红灯,*SY*:边道黄灯,*SG*:边道绿灯)

图6.66 组合逻辑的框图

状态译码器 对时序逻辑中的2位格雷码进行译码,确定系统处在4种状态中的哪一种状态。

灯输出逻辑 使用译码后的状态为主干道和边道的灯部件把相应的交通灯点亮。

触发器逻辑 使用译码后的状态产生信号,从而对长定时器和短定时器进行恰当的初始化(触发)。

6.11.5 组合逻辑的实现

译码逻辑的实现　状态译码器部分有两个输入端(2 位格雷码)和表示 4 种状态中每一种的输出,如图 6.67 所示。两个格雷码输出定义为 S_0 和 S_1,4 种状态输出标注为 SO_1、SO_2、SO_3 和 SO_4。状态输出的布尔表达式如下:

$SO_1 = \overline{S}_1\overline{S}_0$

$SO_2 = \overline{S}_1 S_0$

$SO_3 = S_1 S_0$

$SO_4 = S_1\overline{S}_0$

状态译码器逻辑的真值表如表 6.11 所示。

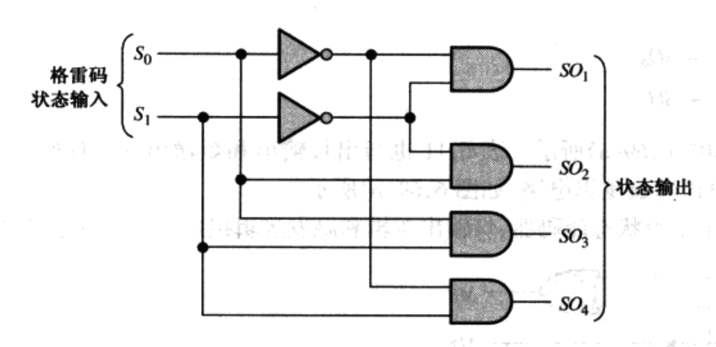


图 6.67　状态译码器的逻辑电路

表 6.11　组合逻辑的真值表

状态输入		状态输出				灯输出						触发输出	
S_1	S_0	SO_1	SO_2	SO_3	SO_4	MR	MY	MG	SR	SY	SG	长	短
0	0	1	0	0	0	0	0	1	1	0	0	1	0
0	1	0	1	0	0	0	1	0	1	0	0	0	1
1	1	0	0	1	0	1	0	0	0	0	1	1	0
1	0	0	0	0	1	1	0	0	0	1	0	0	1

注:(状态输出高电平有效,灯输出高电平有效。*MR*:主干道红灯,*MY*:主干道黄灯,*MG*:主干道绿灯;*SR*:边道红灯,*SY*:边道黄灯,*SG*:边道绿灯)

6.11.6 灯输出逻辑的实现

灯输出逻辑需要四种状态输出,并且产生六个输出用来驱动交通灯。这些输出定义为 *MR*、*MY*、*MG*(主干道红灯,主干道黄灯,主干道绿灯)和 *SR*、*SY*、*SG*(边道红灯,边道黄灯,边道绿灯)。参见真值表 6.11,可以知道交通灯输出可以表示为

$MR = SO_3 + SO_4$

$MY = SO_2$

$MG = SO_1$

$SR = SO_1 + SO_2$

$SY = SO_4$

$SG = SO_3$

输出逻辑的实现如图 6.68 所示。

6.11.7 触发逻辑的实现

触发逻辑产生两个输出。当系统处在第一种状态（00）或第三种状态（11）时，长输出的低电平到高电平的变化触发 25 秒定时电路。当系统进入第二种状态（01）或第四种状态（10）时，短输出电平的低到高的变化触发 4 秒定时器电路。触发输出如表 6.11 所示，等式如下：

长触发 $= SO_1 + SO_3$

短触发 $= SO_2 + SO_4$

触发逻辑如图 6.69(a)所示。表 6.11 也指出长输出和短输出互为反码，因此一个或门和一个反相器就可以完成逻辑电路，如图 6.69(b)所示。

图 6.70 给出了由状态译码器、灯输出逻辑和触发逻辑组成的完整的组合逻辑电路。

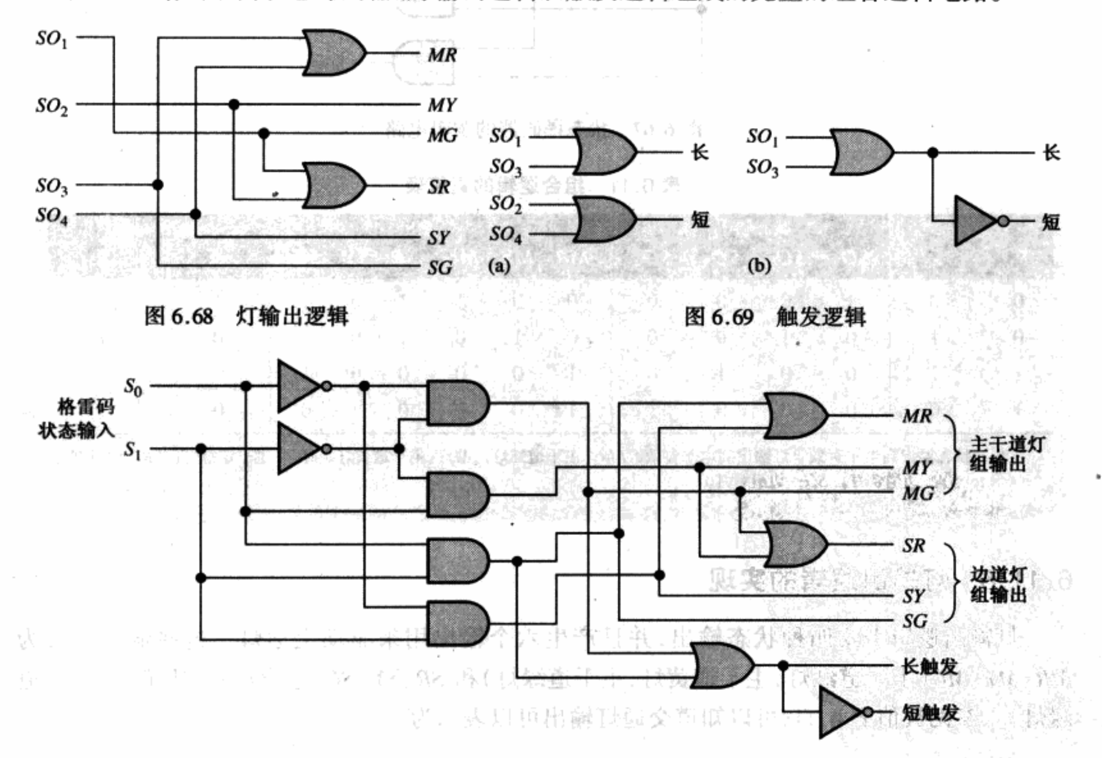


图 6.68 灯输出逻辑

图 6.69 触发逻辑

图 6.70 完整的组合逻辑电路

6.11.8 系统任务

- **任务 1**:在组合逻辑电路的输入 S_0 和 S_1 加上 2 位格雷码波形信号,并且画出所有的输出波形。
- **任务 2**:给出如何用 74×× 芯片完成组合逻辑电路。
- **可选任务**:写出描述组合电路的 VHDL 程序。

自测题 (答案在本章的结尾)

1. 一个半加器的特点是
   (a)有 2 个输入和 2 个输出　(b)有 3 个输入和 2 个输出
   (c)有 2 个输入和 3 个输出　(d)有 2 个输入和 1 个输出
2. 一个全加器的特点是
   (a)有 2 个输入和 2 个输出　(b)有 3 个输入和 2 个输出
   (c)有 2 个输入和 3 个输出　(d)有 2 个输入和 1 个输出
3. 一个全加器的输入为 $A=1, B=1, C_{in}=0$。则其输出为
   (a)$\sum=1, C_{out}=1$　(b)$\sum=1, C_{out}=0$
   (c)$\sum=0, C_{out}=1$　(d)$\sum=0, C_{out}=0$
4. 一个 4 位并行加法器可以用于相加
   (a)两个 4 位二进制数　(b)两个 2 位二进制数
   (c)一次 4 位　(d)顺序相加 4 个位
5. 如果将一个 4 位并行加法器扩展为一个 8 位加法器,必须
   (a) 使用四个 4 位的互不相连的加法器
   (b) 使用两个 4 位加法器,并将其中一个加法器的和输出与另一个加法器的进位输入相连
   (c) 使用八个 4 位不相连的加法器
   (d) 使用两个 4 位加法器,并将其中一个加法器的进位输出与另一个加法器的进位输入相连
6. 如果一个 74HC85 比较器的输入为 $A=1011, B=1001$,则其输出为
   (a)$A>B=0, A<B=1, A=B=0$　(b) $A>B=1, A<B=0, A=B=0$
   (c)$A>B=1, A<B=1, A=B=0$　(d) $A>B=0, A<B=0, A=B=1$
7. 一个输出为低电平有效的 16 选 1 译码器,在它的十进制 12 输出中出现了一个低电平,求这个译码器的输入是什么?
   (a)$A_3A_2A_1A_0=1010$　(b)$A_3A_2A_1A_0=1110$　(c)$A_3A_2A_1A_0=1100$　(d)$A_3A_2A_1A_0=0100$
8. 一个 BCD-7 段译码器的输入为 0100,则其有效输出为
   (a)a, c, f, g　(b)b, c, f, g　(c)b, c, e, f　(d)b, d, e, g
9. 如果一个八进制-二进制优先编码器的 0,2,5,6 输入都为有效电平,则其高电平有效的二进制输出为
   (a)110　(b)010　(c)101　(d) 000
10. 在一般情况下,一个多路复用器有
    (a)一个数据输入、几个数据输出和几个选择输入
    (b)一个数据输入、一个数据输出和一个选择输入
    (c)几个数据输入、几个数据输出和几个选择输入
    (d)几个数据输入、一个数据输出和几个选择输入

11. 数据选择器与下列哪种器件基本相同?

    (a)译码器 (b)多路分配器 (c)多路复用器 (d)编码器

12. 下列代码中,哪一个代码含有偶校验?

    (a)10011000 (b)01111000 (c)11111111

    (d)11010101 (e)以上所有选项 (f)选项(b)和(c)

习题

6.1节 基本加法器

1. 如图6.4所示的全加器,如果这个全加器的输入为下列选项,请确定每个输出门的逻辑状态(1或0):

   (a)$A=1, B=1, C_{in}=1$ (b)$A=0, B=1, C_{in}=1$ (c)$A=0, B=1, C_{in}=0$

2. 如果下列选项为一个全加器所产生的输出,则其输入是

   (a)$\Sigma=0, C_{out}=0$ (b)$\Sigma=1, C_{out}=0$

   (c)$\Sigma=1, C_{out}=1$ (d)$\Sigma=0, C_{out}=1$

3. 如果一个全加器的输入为下列选项,请确定这个全加器的输出:

   (a)$A=1, B=0, C_{in}=0$ (b)$A=0, B=0, C_{in}=1$

   (c)$A=0, B=1, C_{in}=1$ (d)$A=1, B=1, C_{in}=1$

6.2节 并行二进制加法器

4. 并行加法器如图6.71所示,请根据电路中的逻辑操作分析来确定它的和,使用两个输入数字的手算的方法检验其结果。

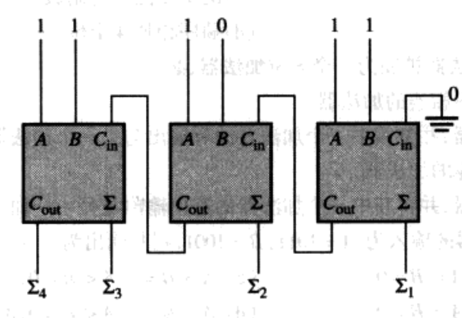


图6.71

5. 如图6.72所示的电路及输入,重复习题1。

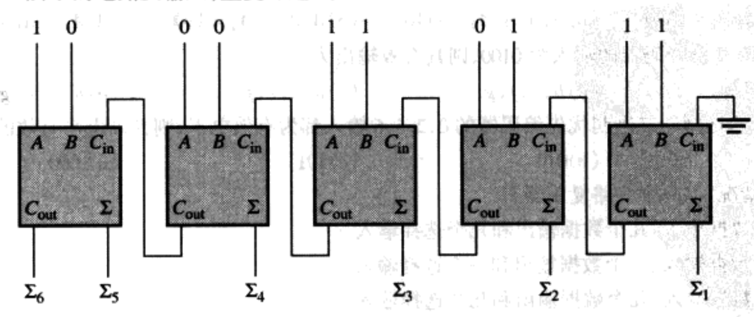


图6.72

6. 一个2位加法器的输入波形如图6.73所示。建立一个时序图，并根据这个时序图中的输入，确定和的波形及进位输出波形。

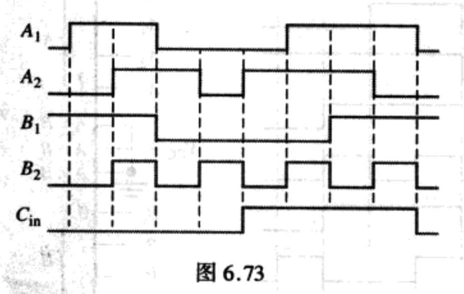


图6.73

7. 4位并行加法器输入位的顺序(最右位第一)如下，确定每个和输出中的位顺序。

A_1	1001
A_2	1110
A_3	0000
A_4	1011
B_1	1111
B_2	1100
B_3	1010
B_4	0010

8. 在检测一个74LS283 4位并行加法器的过程中，其引脚的电压电平如下列所示：1-高，2-高，3-高，4-高，5-低，6-低，7-低，9-高，10-低，11-高，12-低，13-高，14-高，15-高。请确定这个集成电路是否正常工作？

6.3节 异步进位与超前进位加法器

9. 在8位并行异步进位加法器中的八个全加器的每一个有下面的传输延迟：

A 到Σ和 C_{out}：40 ns

B 到Σ和 C_{out}：40 ns

C_{in}到Σ：35 ns

C_{in}到 C_{out}：25 ns

为两个8位数的加法确定其最大的全部运算时间。

10. 给出使得图6.18的4位超前加法器变成5位加法器的附加逻辑电路。

6.4节 比较器

11. 这个比较器的波形如图6.74所示。请确定这个比较器的输出波形($A = B$)。

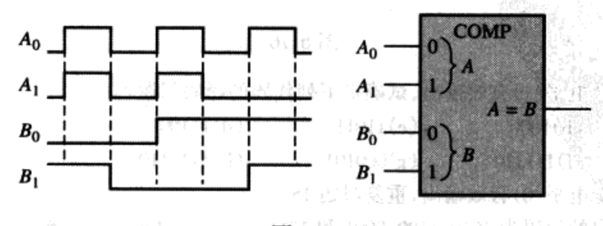


图6.74

12. 一个4位比较器如图6.75所示，请根据所示输入，绘出每一个输出波形。其输出为高电平有效。

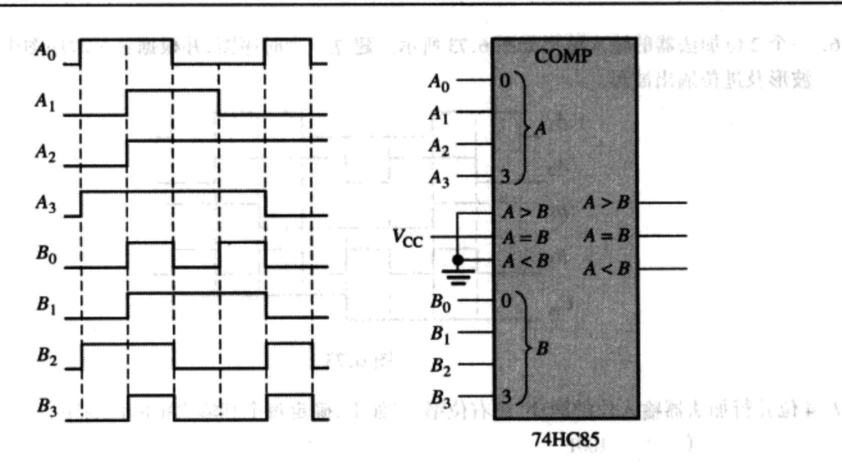


图 6.75

13. 如图 6.22 所示,请根据下列每一组二进制数字,确定比较器的输出状态。

(a) $A_3A_2A_1A_0 = 1100$　$B_3B_2B_1B_0 = 1001$

(b) $A_3A_2A_1A_0 = 1000$　$B_3B_2B_1B_0 = 1011$

(c) $A_3A_2A_1A_0 = 0100$　$B_3B_2B_1B_0 = 0100$

6.5 节　译码器

14. 如图 6.76 所示,译码门的每一个输出都为一个高电平。试确定这个译码门输入的二进制代码,其最高有效位为 A_3。

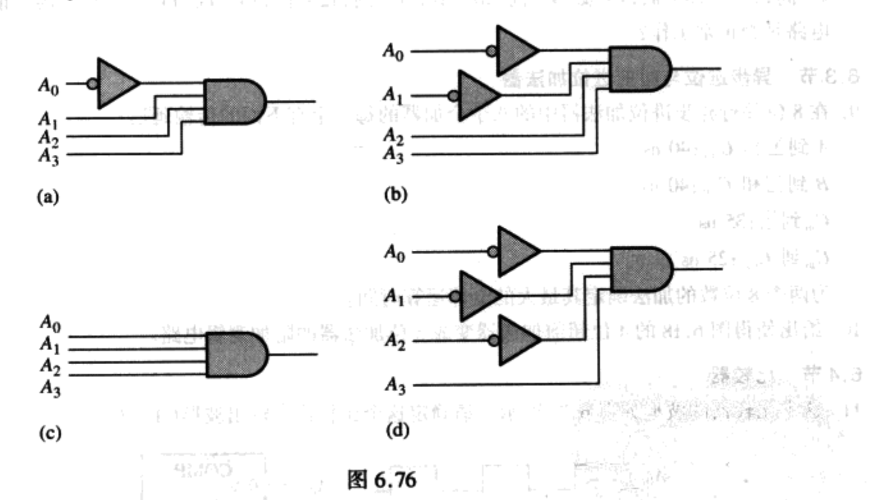


图 6.76

15. 如果需要一个高电平(1)有效输出,试确定下列代码的译码门逻辑:

(a)1101　(b)1000　(c)11011　(d)11100

(e)101010　(f)111110　(g)000101　(h)1110110

16. 如果需要一个低电平(0)有效输出,重复习题 15。

17. 当只希望检测出的代码为 1010、1100、0001 和 1011。用一个高电平有效输出来表示它们的出现。使用一个单一输出,绘制这个最小化的译码逻辑,显示其中任何一个出现在输入的代码。如果为其他代码,输出必须为低电平。

18. 图6.77所示为加在译码逻辑电路的输入波形，请根据输入波形绘出其输出的波形。

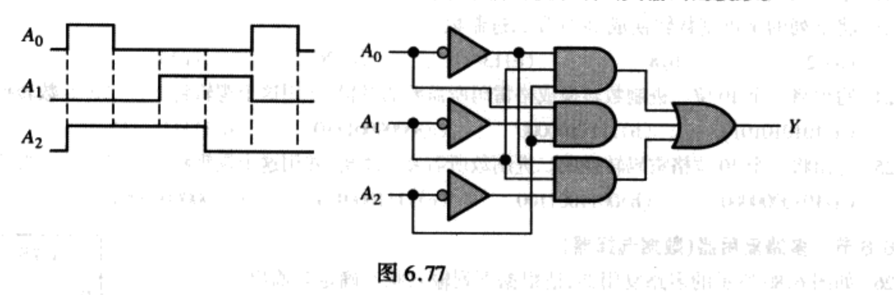


图6.77

19. 如图6.78所示，BCD数按照顺序加到BCD－十进制译码器的输入。请绘出一幅时序图，显示输出之间的关系及每个输出与输入的关系。

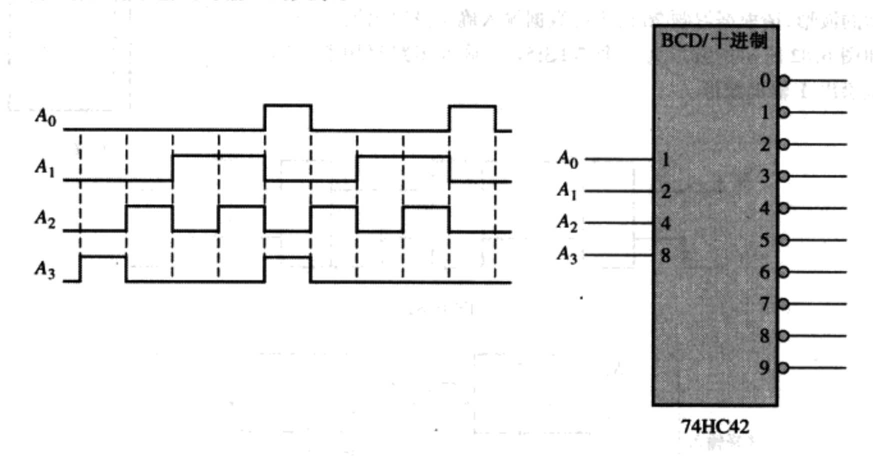


图6.78

20. 一个7段译码器/驱动器的显示如图6.79所示。如果图中所示波形加在芯片的输入，请确定显示器显示的数字的顺序。

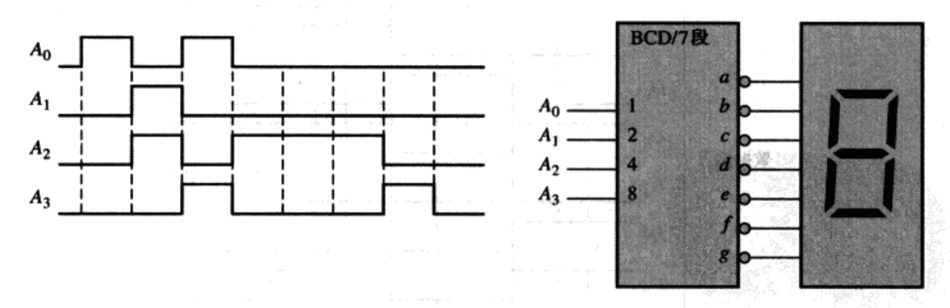


图6.79

6.6节　编码器

21. 十进制－BCD编码器的逻辑如图6.38所示。假设输入9和输入3都为高电平。其输出代码是什么？是否为一个有效的BCD(8421)代码？

22. 一个74HC147编码器的引脚2、5和12都为低电平。如果其他的输入都为高电平，其输出产生的BCD代码是什么？

6.7节 代码转换器

23. 将下列的十进制数转换成BCD和二进制数。

(a) 2　(b)8　(c)13　(d)26　(e)33

24. 写出将一个10位二进制数转换成格雷码所需要的逻辑,并用这个逻辑将下列二进制数转换成格雷码:

(a)1010101010　(b)1111100000　(c)0000001110　(d)1111111111

25. 写出将一个10位格雷码转换成二进制数所需要的逻辑,并用这个逻辑将下列格雷码转换成二进制数:

(a)1010000000　(b)0011001100　(c)1111000111　(d)0000000001

6.8节 多路复用器(数据选择器)

26. 如图6.80所示的多路复用器,请根据下列输入状态确定其输出:

$D_0=0$, $D_1=1$, $D_2=1$, $D_3=0$, $S_0=1$, $S_1=0$

27. 如图6.81所示,如果这个多路复用器的数据选择输入如图6.80所示的波形,请根据习题26给出的数据输入确定其输出波形。

28. 如图6.82所示的波形为一个74LS151八输入多路复用器的输入。试绘出Y输出波形。

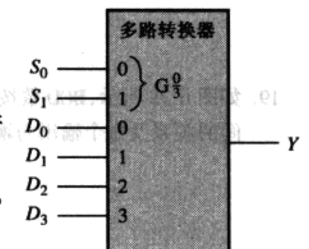


图6.80

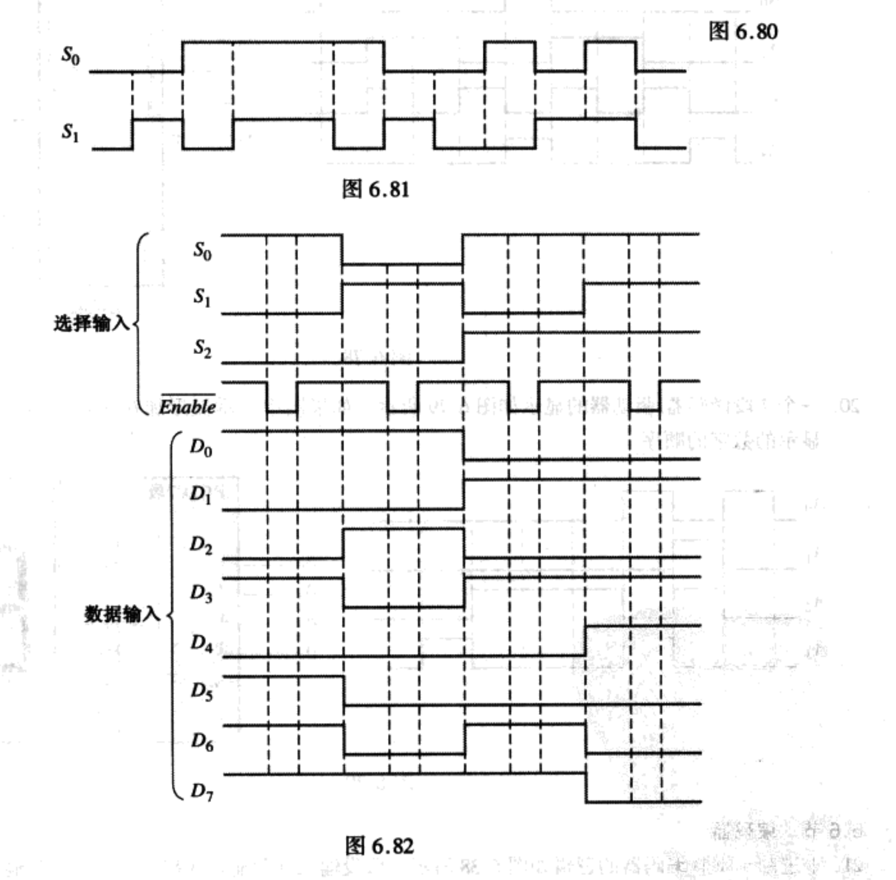


图6.82

6.9 多路分配器

29. 使用74HC154对如下输入进行多路分配的应用,绘制总的时序图(输入和输出):数据选择输入从0000开始按连续的二进制计数顺序循环变化,数据输入为一个串行数据流,携带着表示十进制数

2468 的 BCD 数据。最低有效位(8)在顺序的第一位，由于最低有效位处在第一位，所以它是 4 位输出的第一位。

6.10 奇偶发生器/校验器

30. 4 位奇偶校验逻辑的波形如图 6.83 所示。请根据其输入，确定输出波形。当偶校验发生时，应有多少个位时间，如何得出这个位时间？这个时序图含有 8 个位时间。

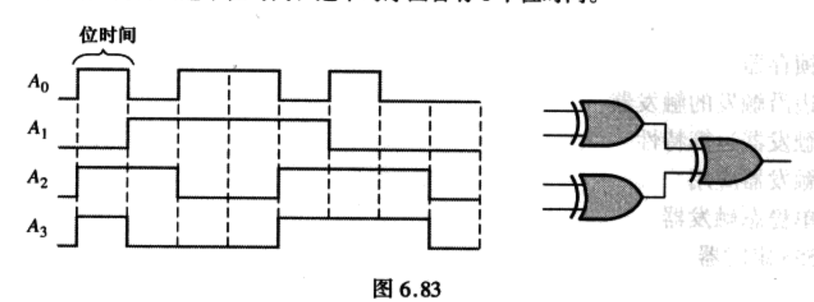


图 6.83

31. 74LS280 9 位奇偶发生器/校验器如图 6.84 所示。请根据这个奇偶发生器/校验器的输入，确定其∑偶数输出和∑奇数输出。参见图 6.59 中的真值表。

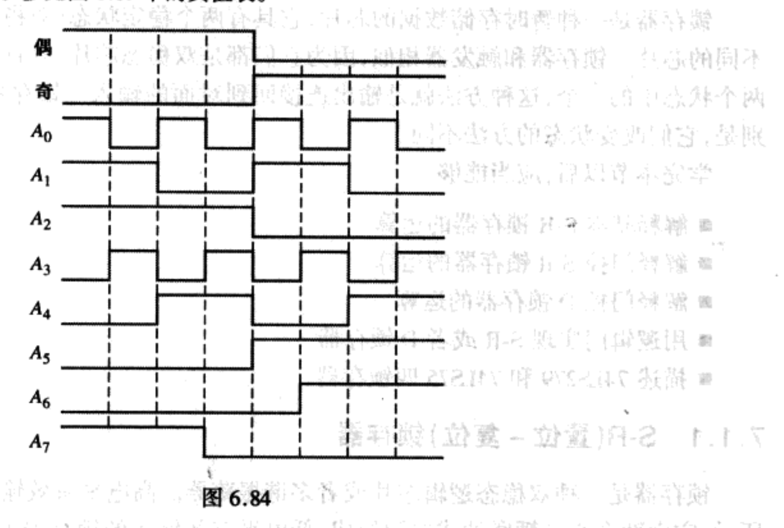


图 6.84

自测题答案

1.(a) 2.(b) 3.(c) 4.(a) 5.(d) 6.(b) 7.(c) 8.(b) 9.(a) 10.(d) 11.(c) 12.(f)

第7章　锁存器、触发器和定时器

章节提纲

7.1　锁存器

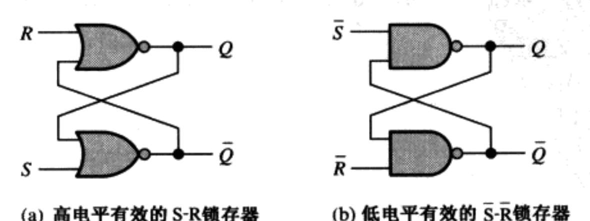
锁存器是一种暂时存储数据的芯片，它具有两个稳定状态（双稳态），一般归类于和触发器不同的芯片。锁存器和触发器相似，因为它们都是双稳态芯片，并且通过反馈方法可以稳定在两个状态中的一个，这种方法就是输出连接回到对面的输入。锁存器和触发器之间的主要区别是，它们改变状态的方法不同。

学完本节以后，应当能够

- 解释基本S-R锁存器的运算
- 解释门控S-R锁存器的运算
- 解释门控D锁存器的运算
- 用逻辑门实现S-R或者D锁存器
- 描述74LS279和74LS75四锁存器

7.1.1　S-R（置位－复位）锁存器

锁存器是一种双稳态逻辑芯片或者多谐振荡器。高电平有效输入S-R锁存器由图7.1(a)所示，它由两个交叉耦联的或非门组成；低电平有效输入的锁存器由图7.1(b)所示，它由两个交叉耦联的与非门组成。注意每个门的输出都连接到对面门的输入上。这就产生了正反馈，这是所有锁存器和触发器的特征。



(a) 高电平有效的S-R锁存器　　(b) 低电平有效的$\bar{S}$-$\bar{R}$锁存器

图7.1　S-R锁存器的两个版本



计算机小知识

锁存器有时候在计算机系统中用来向总线上多路传输数据。例如，来自外部数据源输入计算机的数据不得不和其他的数据源共享数据总线。当数据总线对于外部数据源变得不可用时，现存的数据就必须临时存储起来，而安置在外部数据源和数据总线之间的锁存器就可以执行这个任务。当数据总线对于外部数据源不可用时，锁存器就必须从总线上断开，使用一种称为三态的方法。当数据总线可用时，外部数据就会通过锁存器，因此称为透明锁存器。门控 D 锁存器完成这个功能，因为当它开启时，其输入中的数据就会出现在输出上，就像直接连接一样。一旦锁存器关闭，输入中的数据就存储起来了。

为了解释锁存器的运算，使用图 7.1(b)中的与非门锁存器。使用用于表示与非门的非 - 或等价符号，把这个锁存器重新绘制在图 7.2 中。这样做是因为在 $\bar{S}$ 和 $\bar{R}$ 线上是低电平有效输入。

图 7.2 中的锁存器具有两个输入 $\bar{S}$ 和 $\bar{R}$，以及两个输出 Q 和 $\bar{Q}$。假设这两个输入和 Q 输出都是高电平。由于 Q 输出连接回到门 G_2 的一个输入上，而 $\bar{R}$ 输入为高电平，所以 G_2 的输出必定为低电平。这个低电平输出回接耦联到门 G_1 的一个输入上，以保证它的输出为高电平。

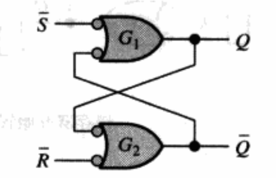


图 7.2　图 7.1(b)中的与非门锁存器的非 - 或等价图

◇ 锁存器可以保持在两种状态的任何一种状态下，即为置位或复位。

当 Q 输出为高电压时，锁存器就处于置位状态。这个状态将一直保持下去直到低电平短暂地加在 $\bar{R}$ 输入上。在 $\bar{R}$ 输入为低电平、$\bar{S}$ 输入为高电平时，门 G_2 的输出就被迫为高电平。位于 $\bar{Q}$ 输出上的这个高电平回接耦联到 G_1 的一个输入上，而 $\bar{S}$ 输入为高电平，所以 G_1 的输出就变为低电平。然后，Q 输出上的这个低电平又回接耦联到 G_2 的一个输入上，从而保证 $\bar{Q}$ 输出保持为高电平，即使 $\bar{R}$ 输入上的低电平移走，Q 输出仍然为高电平。当 Q 输出为低电平时，锁存器就处于复位状态。这时锁存器将一直保持在复位状态直至低电平加在 $\bar{S}$ 输入上。

在一般的运算中，锁存器的输出总是互为反相的。

◇ 置位(SET)的意思是 Q 输出为高电平。

◇ 复位(RESET)的意思是 Q 输出为低电平。

当 Q 为高电平时，$\bar{Q}$ 为低电平，而当 Q 为低电平时，$\bar{Q}$ 为高电平。

当低电平同时加在 $\bar{S}$ 和 $\bar{R}$ 时，低电平有效输入 $\bar{S}$-$\bar{R}$ 锁存器的运算中就会发生无效情况。只要低电平是同时加在输入上，Q 和 $\bar{Q}$ 输出就被迫为高电平，因此违反了输出基本上互为反相的运算。同样，如果低电平同时被释放，两个输出将趋向于变为低电平。由于门的传输延迟时间总会有一些小差别，因此其中有一个门在转变中占据优势，它的 1 输出变为 0。反过来，这就会迫使较慢门的输出保持为高电平。在这种情况下，就不能准确地预知锁存器的下一个状态。

图 7.3 解释了在输入电平的 4 种可能组合的每一种情况下，低电平有效输入 $\bar{S}$-$\bar{R}$ 锁存器的运算(前三个组合是有效的，但是最后一个是无效的)。表 7.1 以真值表的形式总结了逻辑运算。图 7.1(a)中高电平有效输入的或非门的运算也一样，但是需要使用相反的逻辑电平。

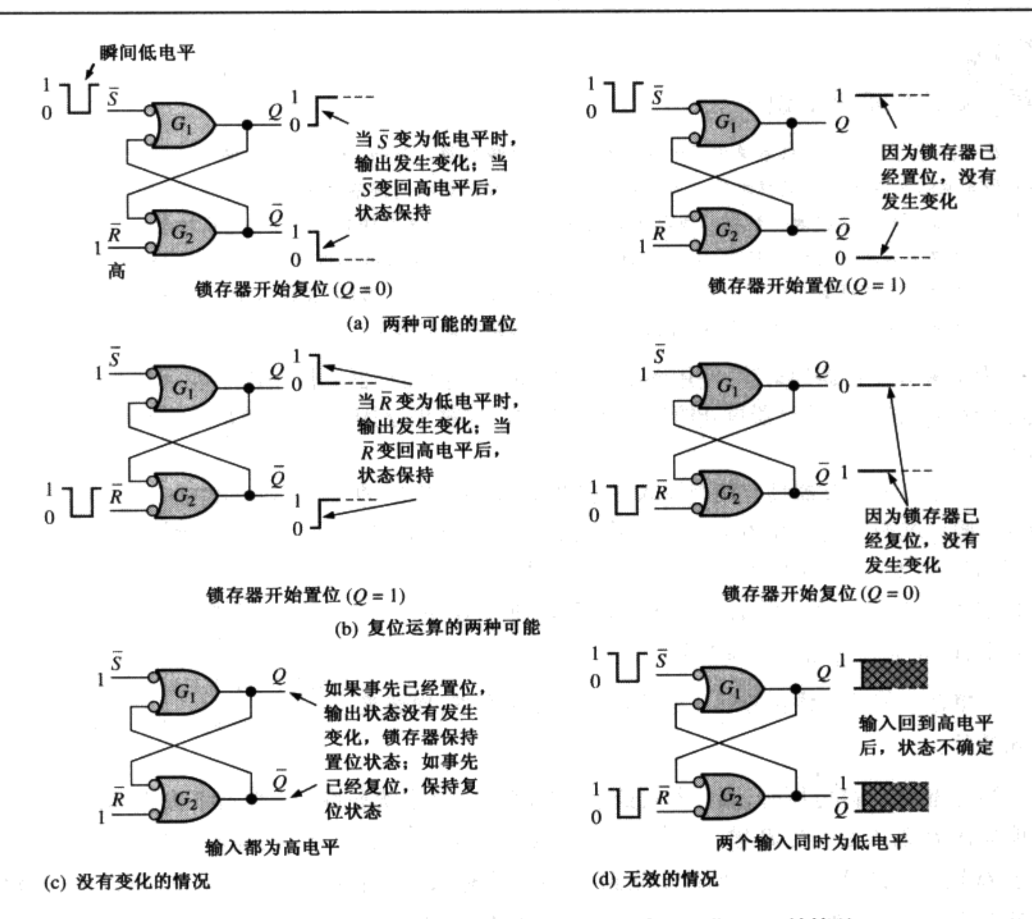


图 7.3 锁存器运算的三种模式(置位,复位,没有变化)和无效情况

表 7.1 低电压有效输入 $\bar{S}$-$\bar{R}$ 锁存器的真值表

输入		输出		说明
$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	
1	1	NC	NC	没有变化，锁存器保持当前的状态
0	1	1	0	锁存器置位
1	0	0	1	锁存器复位
0	0	1	1	无效情况

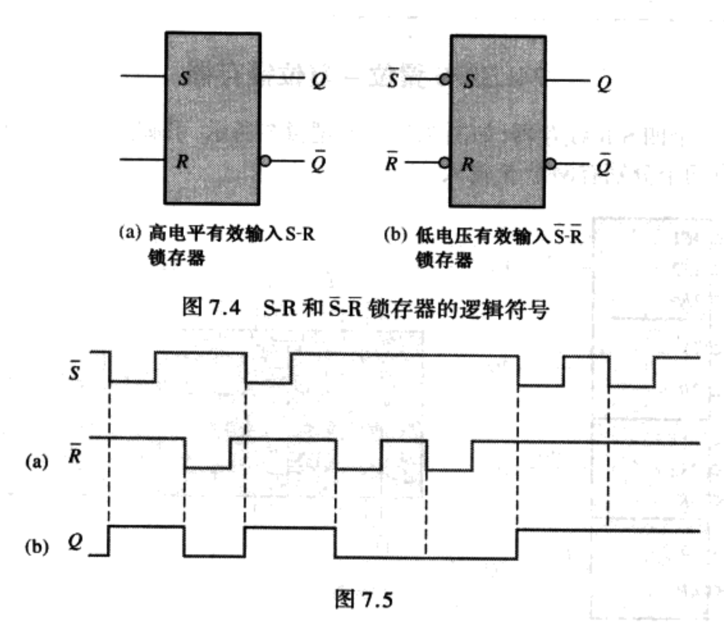
高电平有效输入和低电平有效输入锁存器的逻辑符号如图 7.4 所示。

例 7.1 解释了低电平有效输入 $\bar{S}$-$\bar{R}$ 锁存器如何对输入上的情况做出响应。低电平脉冲以某种顺序加到每个输入上,结果观察到 Q 的输出波形。避免 $\bar{S}=0$、$\bar{R}=0$ 的情况,因为这导致了一个无效的运算模式,并且这是任何置位－复位锁存器的一个主要缺陷。

例 7.1 如果图 7.5(a)的 $\bar{S}$ 和 $\bar{R}$ 波形应用于图 7.4(b)中锁存器的输入上,确定在 Q 输出上将会观察到的波形。假设 Q 的初始值为低电平。

解: 参见图 7.5(b)。

相关问题: 如果图 7.5(a)被反相后再加在输入上,确定高电平有效输入 S-R 锁存器的 Q 输出。



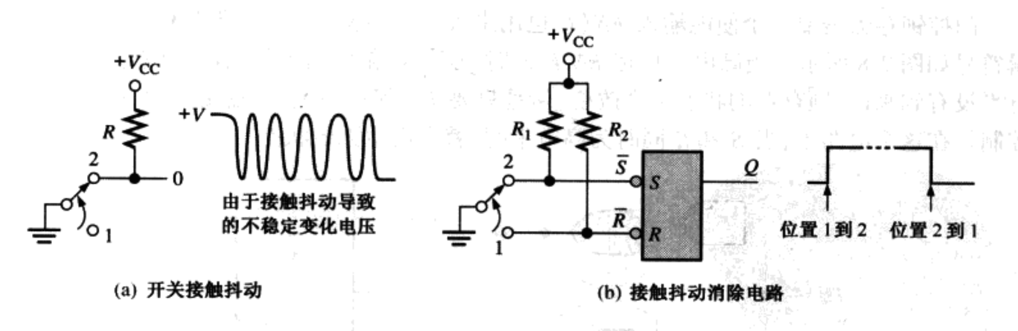
(a) 高电平有效输入S-R锁存器　(b) 低电压有效输入$\bar{S}$-$\bar{R}$锁存器

图 7.4　S-R 和 $\bar{S}$-$\bar{R}$ 锁存器的逻辑符号

图 7.5

7.1.2 应用举例

锁存器作为触点抖动消除器　$\bar{S}$-$\bar{R}$ 锁存器的一个很好的应用例子是消除机械开关接触"抖动"。当开关的触点和开关闭合处的接触面撞击时,会发生几次物理振动或抖动,然后才能形成最后的固定接触。虽然这些抖动的持续时间很短,但是它们产生电压尖脉冲,这些电压尖脉冲在数字系统中常常是不可接受的。这种情况如图 7.6(a)所示。



(a) 开关接触抖动　(b) 接触抖动消除电路

图 7.6　锁存器用以消除开关的接触抖动

$\bar{S}$-$\bar{R}$ 锁存器可以用来消除开关抖动的影响,如图 7.6(b)所示。这个开关一般处在位置 1,保持 $\bar{R}$ 输入为低电平和锁存器复位。当开关合向位置 2 时,由于上拉电阻连接 V_{CC},$\bar{R}$ 就变为高电平,开关闭合的第一次接触,$\bar{S}$ 变为低电平。尽管在开关抖动之前,$\bar{S}$ 在低电平上仅仅保持了很短的时间,但是这点时间足以使锁存器置位。此后,由于开关抖动在 $\bar{S}$ 输入上产生的

任何电压尖脉冲不会影响锁存器,因此保持为置位状态。注意锁存器的 Q 输出提供了从低电平到高电平的净变化,因此消除了由于触点抖动而产生的电压尖脉冲。类似地,当开关拨回到位置 1 时,就会产生从高电平到低电平的净变化。

74LS279 置位 – 复位锁存器

74LS279 是一个四 $\overline{S}$-$\overline{R}$ 锁存器,如图 7.7(a)的逻辑图所示,引脚图如图 7.7(b)所示。注意四个锁存器中有两个分别有两个 $\overline{S}$ 输入。

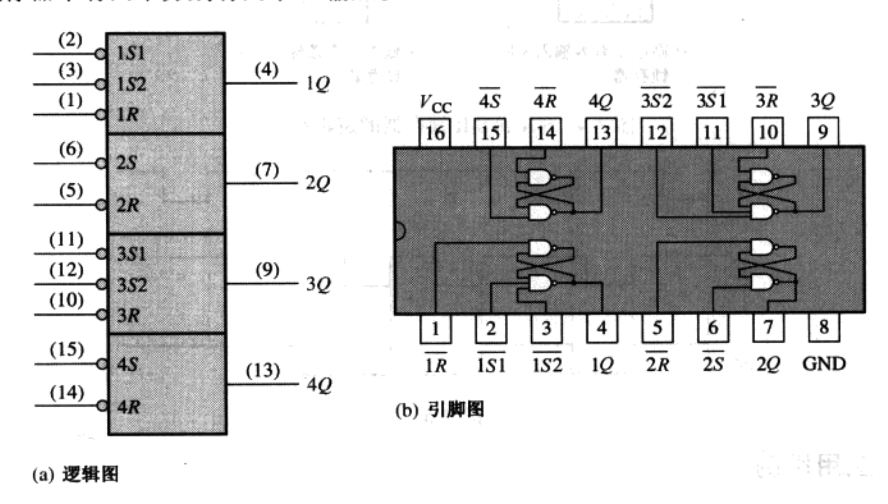


图 7.7 74LS279 四 $\overline{S}$-$\overline{R}$ 锁存器

7.1.3 门控 S-R 锁存器

门控锁存器需要一个使能输入 EN(G 也用来表示使能输入)。门控锁存器的逻辑图和逻辑符号如图 7.8 所示。当高电平加在 EN 输入时,S 和 R 输入控制锁存器的状态。在 EN 的高电平没有到来前,锁存器的状态不会改变;但是只要 EN 保持高电平,输出就由 S 和 R 的状态控制。在这个电路中,当 S 和 R 同时为高电平时,就会发生无效状态。

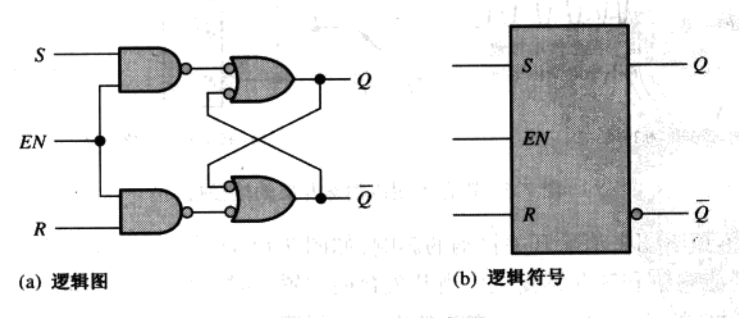


图 7.8 门控 S-R 锁存器

例 7.2 如果图 7.9(a)所示的输入加到初始状态为复位的门控 S-R 锁存器，确定 Q 输出波形。

解：Q 波形如图 7.9(b)所示。当 S 为高电平和 R 为低电平时，EN 上的高电平输入就使锁存器置位。当 S 为低电平和 R 为高电平时，EN 上的高电平输入就使锁存器复位。

相关问题：如图 7.9(a)所示，如果 S 和 R 输入反相，确定门控 S-R 锁存器的 Q 输出。

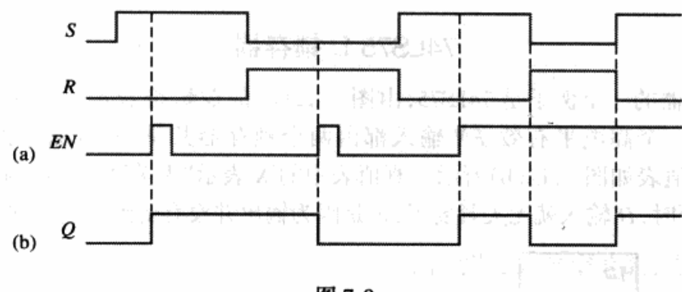


图 7.9

7.1.4 门控 D 锁存器

另一种类型的门控锁存器称为 D 锁存器。它不同于 S-R 锁存器，因为它除了 EN 之外只有一个输入。这个输入称为 D（数据）输入。图 7.10 包括了 D 锁存器的逻辑图和逻辑符号。当 D 输入为高电平和 EN 输入为高电平时，锁存器就会置位。当 D 输入为低电平而 EN 为高电平时，锁存器就会复位。用另外一种方式表述为：当 EN 为高电平时，输出 Q 跟随输入 D。

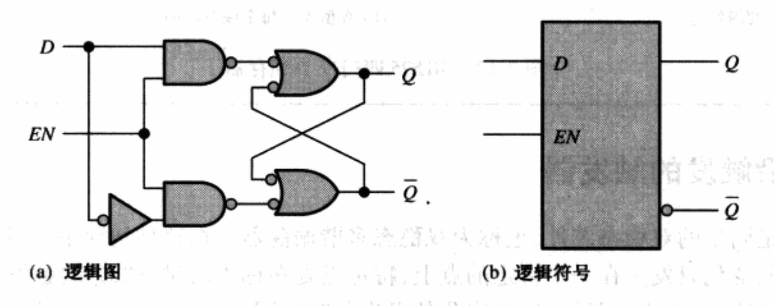


图 7.10 门控 D 锁存器

例 7.3 如果图 7.11(a)中所给出的输入加在初始状态为复位的门控 D 锁存器上，请确定 Q 输出波形。

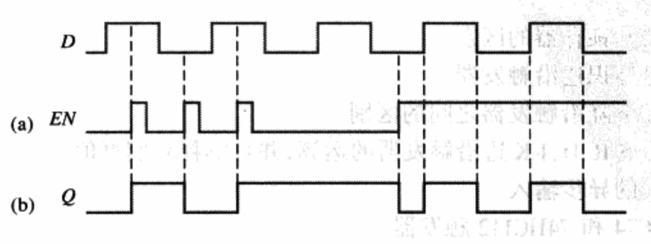


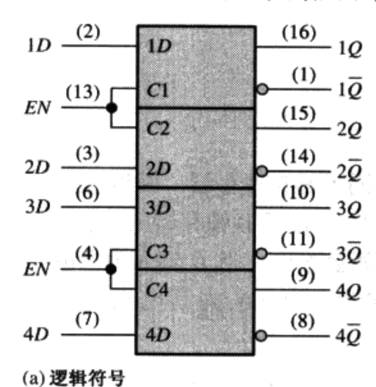
图 7.11

解: Q 波形如图 7.11(b)所示。当 D 为高电平和 EN 为高电平时，Q 就会变为高电平。当 D 为低电平和 EN 为高电平时，Q 就会变为低电平。当 EN 为低电平时，锁存器的状态不受 D 输入的影响。

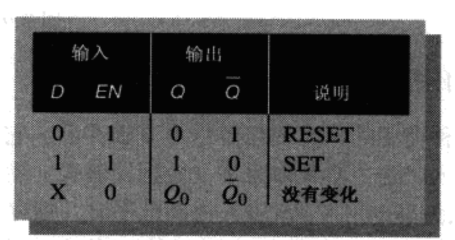
相关问题: 如果图 7.11(a)的 D 输入反相，确定该门控 D 锁存器的 Q 输出。

74LS75 D 锁存器

门控 D 锁存器的一个例子是 74LS75，由图 7.12(a)的逻辑符号表示。这个芯片具有 4 个锁存器。注意每一个高电平有效 EN 输入都由两个锁存器共享，并且命名为控制输入(C)。每个锁存器的真值表如图 7.12(b)所示。真值表中的 X 表示"无关"的情况。在这种情况下，当 EN 输入为低电压时，D 输入就无关紧要了，这是因为输出并没有受到影响，保持先前的状态。



(a) 逻辑符号



输入		输出		
D	EN	Q	$\overline{Q}$	说明
0	1	0	1	RESET
1	1	1	0	SET
X	0	Q_0	$\overline{Q}_0$	没有变化

注：Q_0 是在指示的输入条件建立之前的先前输出电平。

(b) 真值表（每个锁存器的）

图 7.12 74LS75 四门控 D 锁存器

7.2 边沿触发的触发器

触发器是同步的双稳态芯片，也称为双稳态多谐振荡器。在这种情况下，名称同步的意思是输出状态的变化只发生在一个特定的点上，特定点处在称为时钟(CLK)的触发输入上，时钟被指定为控制输入 C；也就是说，输出变化的发生与时钟同步。

学完本节以后，应当能够

- 定义时钟
- 定义边沿触发器
- 解释触发器和锁存器的区别
- 用逻辑符号标识边沿触发器
- 讨论上升和下降沿触发器之间的区别
- 讨论并比较 S-R、D、J-K 边沿触发器的运算，并且解释它们真值表的差别
- 讨论触发器的异步输入
- 描述 74AHC74 和 74HC112 触发器

◇ 动态输入指示器▷的意思是触发器的状态变化仅发生在时钟脉冲的边沿。

边沿触发器(edge-triggered flip-flop)的时钟脉冲的状态变化发生在时钟脉冲的正边沿(上升沿)或者负边沿(下降沿)上，并且只有在时钟的这个转换瞬间才对它的输入做出响应。本节介绍的三种类型的边沿触发的触发器是：S-R，D，J-K。虽然在 IC 系列中是没有 S-R 触发器的，但是它是 D 和 J-K 触发器的基础。所有这些触发器的逻辑符号如图 7.13 所示。注意每一种类型都可以是正边沿触发的(在 C 输入上没有小圆圈)或者是负边沿触发的(在 C 输入上有小圆圈)。通过逻辑符号识别边沿触发器的关键是框内时钟输入(C)上的小三角。这个小三角称为动态输入指示器。

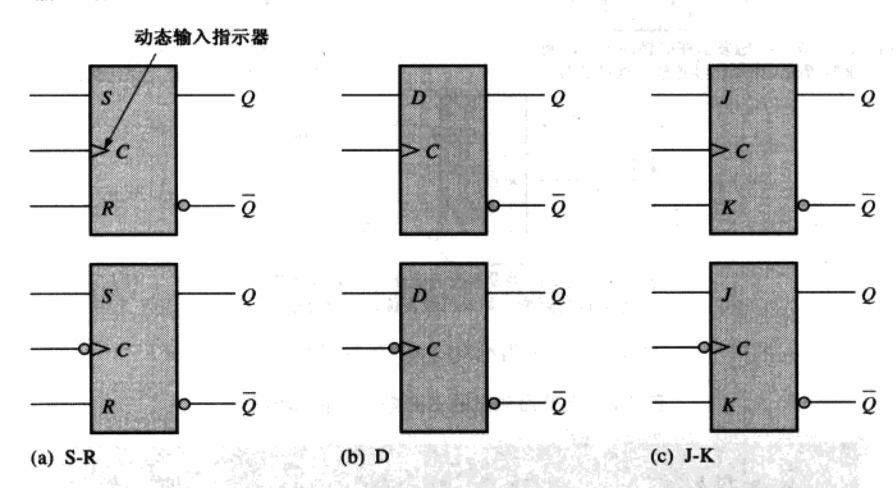


图 7.13　边沿触发器的逻辑符号(顶部：上升沿触发的；底部：下降沿触发的)

7.2.1　边沿触发的 S-R 触发器

◇ S-R 触发器的 S 和 R 输入不能同时为高电平。

S-R 触发器的 S 和 R 输入称为同步输入，因为这两个输入上的数据只能在时钟脉冲的触发边沿传送到触发器的输出。当 S 为高电平和 R 为低电平时，Q 输出在时钟脉冲的触发边沿上变为高电平，且触发器被置位。当 S 为低电平和 R 为高电平时，Q 输出在时钟脉冲的触发边沿上变为低电平，且触发器被复位。当 S 和 R 都是低电平时，输出先前的状态不会改变。当 S 和 R 都是高电平时，就存在无效的情况。

上升沿触发的触发器的基本运算如图 7.14 所示，表 7.2 是这类触发器的真值表。记住，在时钟脉冲的触发边沿之外触发器的状态不能改变。在时钟输入是低电平或高电平时，S 和 R 的输入在任何时间(除了在时钟触发转换瞬间周围的一个很短的时间间隔内)都可以改变，而且不影响输出。

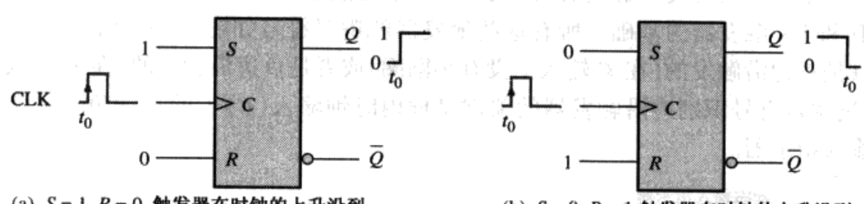
除了时钟脉冲的下降沿是触发边沿之外，下降沿触发的 S-R 触发器的运算和真值表与上升沿触发的芯片一样。



计算机小知识

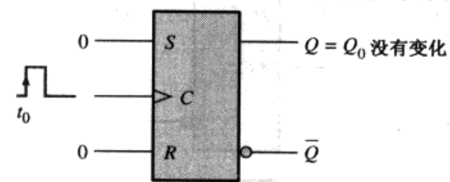
计算机中半导体存储器由大量的独立存储单元组成。每个存储单元都保存一个 1 或者 0。存储器的一种类型是静态随机存储器(SRAM)，其中每个单元都使用触发器，因此只要加上直流电源，触发器可以不确定

地保持为两个状态之一,由此得到静态这个词。这种类型的存储器被归类为易失性存储器,因为当电源断开时,所有存储的数据都会丢失。另一种类型的存储器为动态随机存储器(DRAM),它使用电容而不是触发器作为基本存储元件,并且必须周期性地刷新以保持存储的数据。



(a) $S=1$、$R=0$ 触发器在时钟的上升沿到来时置位(如果已经置位,保持置位)

(b) $S=0$, $R=1$ 触发器在时钟的上升沿到来时复位(如果已经复位,保持复位)



(c) $S=0$, $R=0$ 触发器没有变化(如果已经置位,保持置位;如果已经复位,保持复位)

图 7.14 上升沿触发的 S-R 触发器的运算

表 7.2 上升沿触发的 S-R 触发器的真值表

输入			输出		
S	R	CLK	Q	$\overline{Q}$	说明
0	0	X	Q_0	$\overline{Q}_0$	没有变化
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	?	?	无效

↑——时钟从低电平到高电平的转换

X——不相关("无关")

Q_0——时钟转换以前的输出电平

例 7.4 对于图 7.16(a)中的 S、R 和 CLK 输入,确定图 7.15 中触发器的 Q 和 $\overline{Q}$ 的输出波形。假设该上升沿触发器初始值状态为复位。

解:

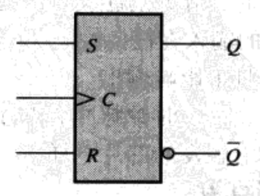


图 7.15

1. 在时钟脉冲 1 处,S 为低电平和 R 为低电平,所以 Q 不会发生变化。
2. 在时钟脉冲 2 处,S 为低电平和 R 为高电平,所以 Q 保持为低电平(复位)。
3. 在时钟脉冲 3 处,S 为高电平和 R 为低电平,所以 Q 变为高电平(置位)。
4. 在时钟脉冲 4 处,S 为低电平和 R 为高电平,所以 Q 变为低电平(复位)。
5. 在时钟脉冲 5 处,S 为高电平和 R 为低电平,所以 Q 变为高电平(置位)。

6. 在时钟脉冲6处，S 为高电平和 R 为低电平，所以 Q 停留在高电平。

一旦 Q 被确定下来，很容易得到 $\overline{Q}$，因为 $\overline{Q}$ 只是 Q 的反码。Q 和 $\overline{Q}$ 的结果波形如图 7.16(b)所示，输入波形在图 7.16(a)中。

相关问题：如果该触发器为下降沿触发的芯片，请为图 7.16(a)中所示的 S 和 R 输入确定 Q 和 $\overline{Q}$。

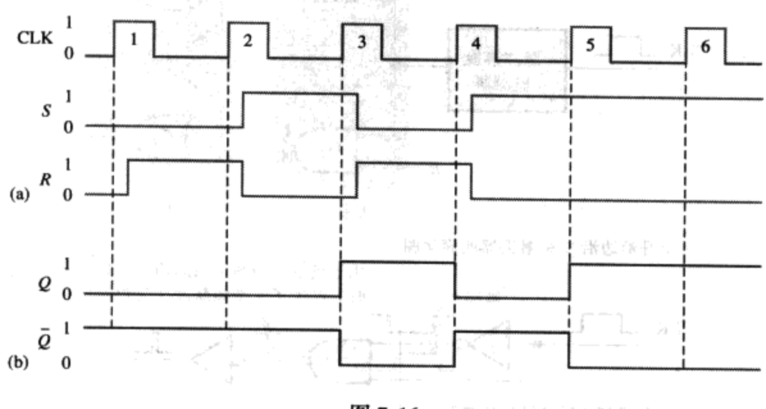


图 7.16

7.2.2 边沿触发的一种方法

边沿触发的 S-R 触发器的简化实现方法如图 7.17(a)所示，该图还用来展示边沿触发的概念。介绍 S-R 触发器并不意味着它是最重要的类型。实际上，D 触发器和 J-K 触发器在 IC 系列中都可以找到，并且比 S-R 类型应用得更加普遍。然而，理解 S-R 触发器是很重要的，这是因为 D 和 J-K 触发器是由 S-R 触发器衍生的。注意，S-R 触发器和门控 S-R 锁存器的唯一区别是 S-R 触发器具有一个脉冲转换检测器。

脉冲转换检测器的一种基本类型如图 7.17(b)所示。正如所见，在到达与非门的一个输入时有一个很短的时间延迟，因此反相后的时钟脉冲到达与非门输入的时间要比原来的时钟脉冲晚几个纳秒。在时钟脉冲的上升沿转换的地方，电路产生了一个持续时间很短的窄脉冲。在下降沿触发的触发器中，时钟脉冲首先被反相，因此就在下降沿处产生一个窄脉冲。

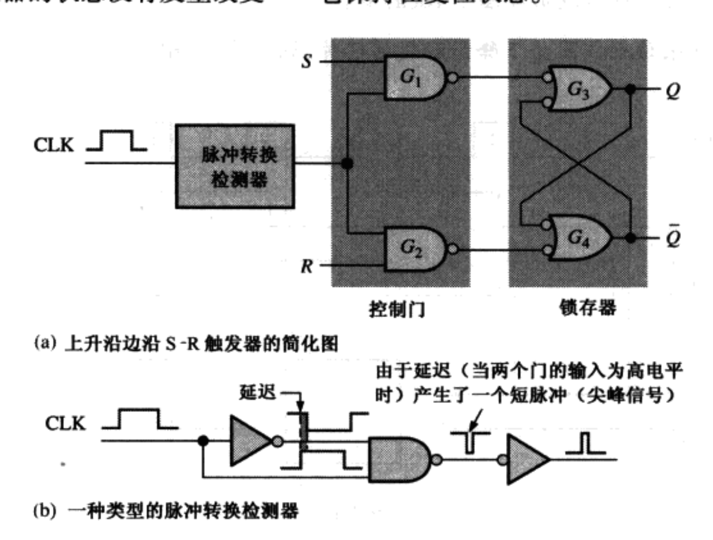


计算机小知识

所有由硬件执行的逻辑运算也可以在软件中实现。例如，J-K 触发器的运算就可以由具体的计算机指令来完成。如果用两个位表示 J 和 K 输入，计算机对于 00 不做任何事情；而对于 10，表示 Q 输出的数据位将会被置位(1)，对于 01，Q 数据位就会被清零(0)，对于 11，Q 数据位将会被求反。虽然使用计算机来模拟触发器可能不常用，但是我们强调的是所有的硬件运算都可以用软件来模拟。

注意，图 7.17 中的电路被分成两部分，一部分被标记为控制门，另一部分被标记为锁存器。控制门引导或者控制时钟窄脉冲进入门 G_3 的输入或者进入门 G_4 的输入，这取决于 S 和 R 输入的状态。为了理解该触发器的运行，开始时假设它处于复位状态($Q=0$)，而 S、R 和 CLK 输入全部为低电平。对于这种情况，门 G_1 和门 G_2 的输出都是高电平。Q 输出上的低电平被耦联回接到门 G_4 的一个输入上，使得其输出为高电平。因为是高电平，G_3 的两个输入都

是高电平(记住,门 G_1 的输出为高电平),这就使得 Q 输出保持为低电平。如果一个脉冲加在 CLK 输入,门 G_1 和 G_2 的输出仍然保持为高电平,这是因为它们被 S 和 R 输入上的低电平禁止了;所以触发器的状态没有发生改变——它保持在复位状态。



(a) 上升沿边沿 S-R 触发器的简化图

(b) 一种类型的脉冲转换检测器

图 7.17 边沿触发

现在使 S 为高电平,R 为低电平,然后加上一个时钟脉冲。由于门 G_1 的 S 输入现在是高电平,并且当 CLK 变为高电平时,门 G_1 的输出会在低电平上持续一个很短的时间(窄脉冲),这就导致了 Q 输出变为高电平。现在门 G_4 的输入都是高电平(记住,因为 R 为低电平,所以门 G_2 的输出为高电平),这就迫使 $\overline{Q}$ 输出为低电平。$\overline{Q}$ 上的这个低电平又被耦联回接到门 G_3 的输入上,这将确保 Q 输出保持高电平。触发器现在的状态就是置位。图 7.18 解释了在这种情况下发生在触发器内部的逻辑电平的变换。

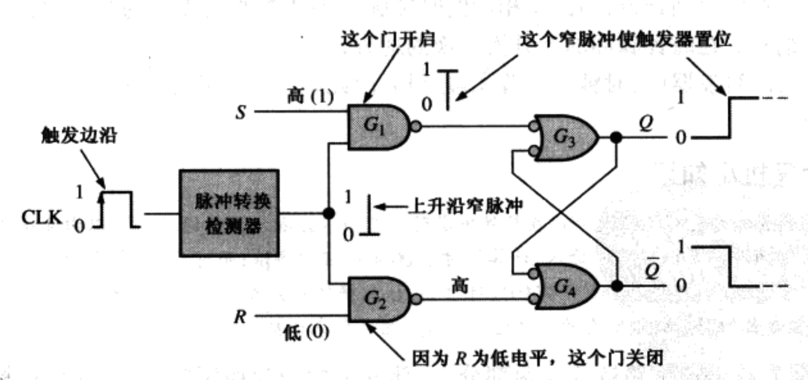


图 7.18 触发器在时钟脉冲的上升沿完成了从复位到置位的转换

下一步,使 S 为低电平,R 为高电平,然后加上一个时钟脉冲。因为 R 输入现在为高电平,而时钟的上升沿在门 G_2 的输出上产生一个下降沿窄脉冲,这就使得 $\overline{Q}$ 输出变换为高电平。由于 $\overline{Q}$ 上的这个高电平,门 G_3 的两个输入现在都成了高电平(记住,由于 S 为低电平,所

以门 G_1 的输出为高电平),这就迫使 Q 输出变为低电平。Q 上的这个低电平被耦联回接到门 G_4 的一个输入上,以确保 $\overline{Q}$ 保持为高电平。现在触发器处于复位状态。

图 7.19 解释了这种情况下发生在触发器内部的逻辑电平的转换。和门控锁存器一样,当 S 和 R 同时是高电平并且有时钟脉冲时,就存在一个无效的情况。这是 S-R 触发器的一个主要缺陷。

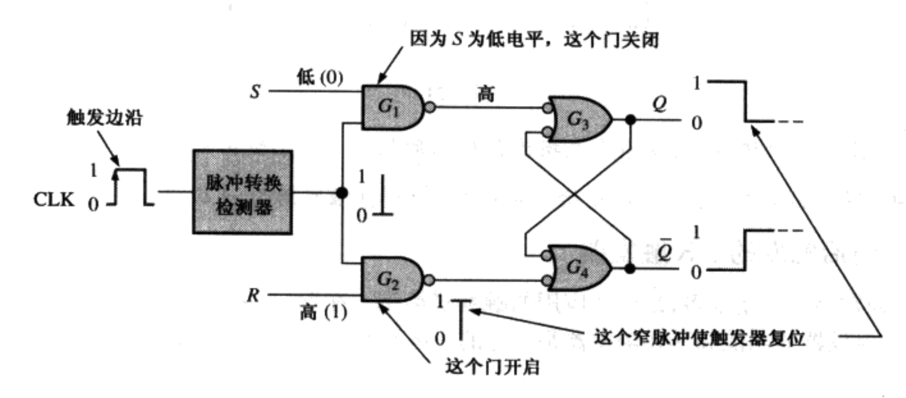


图 7.19 触发器在时钟脉冲的上升沿完成了从置位到复位的转换

7.2.3 边沿触发的 D 触发器

◇ D 触发器的 Q 输出假设了时钟触发边沿的 D 输入状态。

当存储单个数据位(1 或者 0)时,可使用 D 触发器。在 S-R 触发器上加上反相器就形成了基本的 D 触发器,图 7.20 给出了上升沿触发类型。

注意图 7.20 中的触发器除了时钟之外,只有一个输入:D 输入。当时钟脉冲到来时,如果 D 输入上有一个高电平,那么触发器就被置位,这样通过时钟脉冲的上升沿,D 输入上的高电平被触发器存储。当时钟脉冲到来时,如果 D 输入上有一个低电平,那么触发器就被复位,这样通过时钟脉冲的上升沿,D 输入上的低电平被触发器存储。在置位状态下,触发器存储一个 1,而在复位状态下存储一个 0。

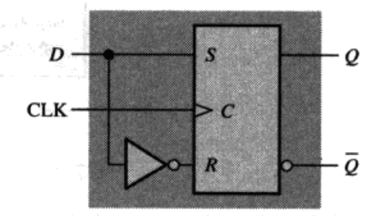


图 7.20 由 S-R 触发器和反相器形成的上升沿D触发器

上升沿触发的 D 触发器的逻辑运算总结于表 7.3 中。当然,下降沿触发芯片的运算是一样的,除了触发发生在时钟脉冲的下降边沿之外。记住,在有效或者触发时钟边沿,Q 跟随 D。

表 7.3 上升沿触发的 D 触发器的真值表

输入		输出		
D	CLK	Q	$\overline{Q}$	说明
1	↑	1	0	SET (存储一个1)
0	↑	0	1	RESET (存储一个0)

↑——时钟转换 低到高

例 7.5 图 7.21(a)中给出 D 输入和时钟的波形，如果触发器初始状态为复位，确定 Q 输出波形。

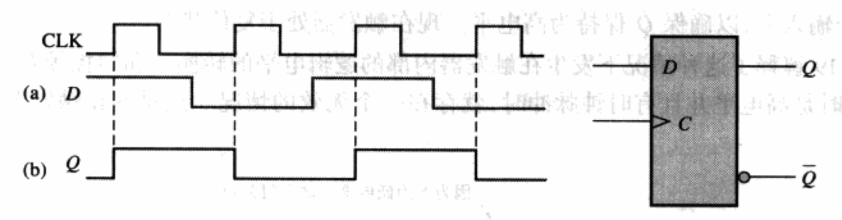


图 7.21

解：在上升时钟边沿上，Q 输出变换为 D 输入的状态。结果输出如图 7.21(b)所示。

相关问题：如果图 7.21(a)中 D 输入被反相，确定 D 触发器的 Q 输出。

7.2.4 边沿触发的 J-K 触发器

J-K 触发器的用途很多，是广泛应用的触发器类型。在置位、复位和运算没有变化情况的方面，J-K 触发器的功能和 S-R 触发器是一样的。区别在于 J-K 触发器没有无效状态而 S-R 触发器有。

图 7.22 给出了上升沿触发的 J-K 触发器的基本内部逻辑。它和 S-R 边沿触发的触发器的区别在于，Q 输出回接到门 G_2 的输入上。而 $\overline{Q}$ 输出回接到门 G_1 的输入。这两个控制输入以 Jack Kilby 的名义标记为 J 和 K(Jack Kilby 发明了集成电路)。J-K 触发器也可以有下降沿触发的类型，这种类型中的时钟输入被反相了。

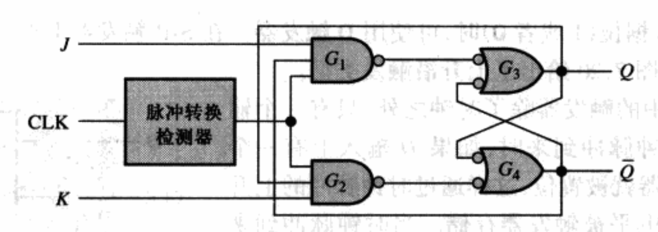


图 7.22 上升沿触发的 J-K 触发器的简化逻辑图

让我们假设图 7.23 中的触发器处于复位状态，并且 J 输入为高电平、K 输入为低电平而不是图中所示。当时钟脉冲到来时，由①所指示的上升沿窄脉冲就会通过门 G_1，这是因为 $\overline{Q}$ 为高电平并且 J 也为高电平。这就会导致触发器中的锁存器部分变为置位状态。

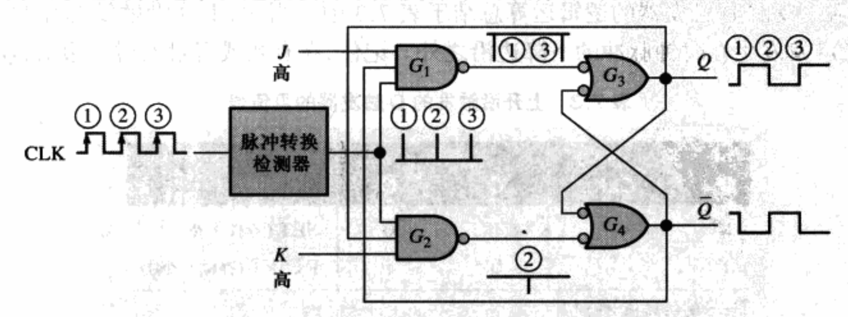


图 7.23 当 $J=1$ 和 $K=1$ 时，解释切换运算下的变换

◇ 在切换模式下，J-K 触发器在每一个时钟脉冲到来时都改变状态。

如果使得 J 为低电平而 K 为高电平，由②所指示的下一个时钟窄脉冲就会通过门 G_2，因为 Q 为高电平而 K 也为高电平。这就会导致触发器中锁存器部分变为复位状态。

如果低电平同时加在 J 和 K 输入上，当时钟脉冲到来时，触发器就会保持当前的状态。J 和 K 上的低电平导致输出没有变化的情况。

到目前为止，在置位、复位及无变化模式方面，J-K 触发器的逻辑运算和 S-R 是一样的。当 J 和 K 输入都是高电平时，运算中的区别显现了。为了说明问题，假设触发器处于复位状态。$\overline{Q}$ 上的高电平使得门 G_1 开启，这样由③所指示的时钟窄脉冲就会通过并使触发器置位。现在 Q 上有一个高电平，这就允许下一个时钟窄脉冲经过门 G_2 并使触发器复位。

正如所见，在每一个相继的时钟脉冲上，触发器变为相反的状态。这个模式称为切换运算。图 7.23 解释了触发器处于切换模式时的变换。连接成切换运算的 J-K 触发器有时候称为 T 触发器。

表 7.4 以真值表的形式总结了边沿触发的 J-K 触发器的逻辑运算。注意这里没有无效状态，而 S-R 触发器有无效状态。除了在时钟脉冲的下降沿触发之外，下降沿触发的芯片的真值表和上升沿触发器的真值表是一样的。

表 7.4　上升沿触发的 J-K 触发器的真值表

输入			输出		
J	K	CLK	Q	$\overline{Q}$	说明
0	0	↑	Q_0	$\overline{Q}_0$	没有改变
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\overline{Q}_0$	Q_0	切换

↑——时钟转换低到高

Q_0——时钟转换以前的输出电平

例 7.6　图 7.24(a)所示的波形加在 J、K 和时钟输入，如图所示。确定 Q 输出，假设该触发器的初始状态为复位。

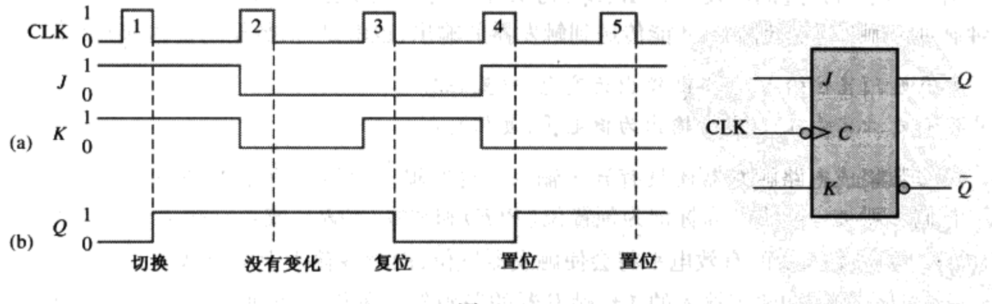


图 7.24

解:

1. 首先,由于这是一个下降沿触发器,如时钟输入上的"小圆圈"所指示的那样,Q 输出仅在时钟脉冲的下降沿改变。
2. 在第一个时钟脉冲上,J 和 K 都是高电平;并且由于这是切换的情况,Q 就变为高电平。
3. 在时钟脉冲 2 上,输入上出现了无变化的情况,就保持 Q 在高电平。
4. 当时钟脉冲 3 到来时,J 为低电平和 K 为高电平,结果就处在复位的情况;Q 变为低电平。
5. 在时钟脉冲 4,J 为高电平和 K 为低电平,结果就处在置位的情况;Q 变为高电平。
6. 当时钟脉冲 5 到来时,J 和 K 仍然处于置位的情况,所以 Q 将保持在高电平。结果 Q 波形如图 7.24(b)所示。

相关问题: 如果图 7.24(a)中的 J 和 K 输入反相,确定 J-K 触发器的 Q 输出。

例 7.7 图 7.25(a)中的波形加在如图所示的触发器上。确定 Q 输出,假设开始于复位状态。

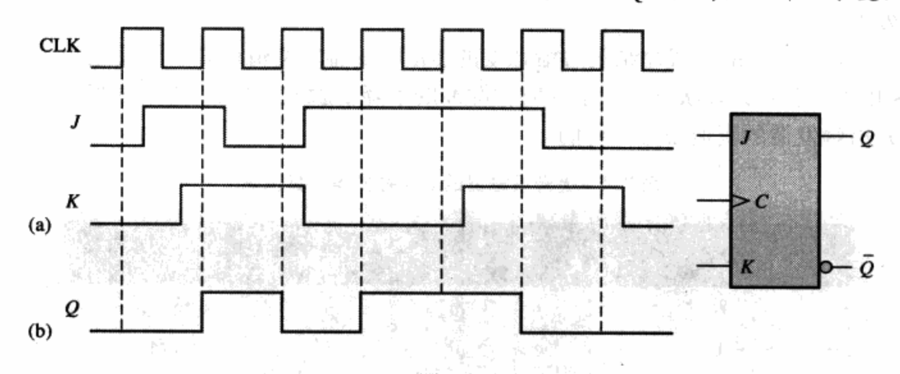


图 7.25

解: 假设 Q 输出的状态由时钟脉冲的上升沿(触发边沿)的 J 和 K 输入状态所确定。在时钟触发边沿到来以后 J 和 K 的变化对输出不会产生影响,如图 7.25(b)所示。

相关问题: 交换 J 和 K 的输入,并确定 Q 输出。

7.2.5 异步预置位输入和清零输入

对于刚刚讨论过的触发器,*S-R*、*D* 和 *J-K* 输入称为同步输入,因为这些输入上的数据仅在时钟脉冲的触发边沿到来时才能传送到触发器的输出;也就是说,数据的传送和时钟同步。

◇ 有效预置位输入使得 Q 输出为高电平(置位)。

◇ 有效清零输入使得 Q 输出为低电平(复位)。

大多数集成电路触发器还具有异步输入。这些输入可以独立于时钟而影响触发器的状态。它们一般被一些生产商标记为预置位(*PRE*)和清零(*CLR*),或者直接置位(S_D)和直接复位(R_D)。预置输入上的有效电平将会使触发器置位,而清零输入上的有效电平将使触发器复位。一个具有预置和清零输入的 J-K 触发器的逻辑符号如图 7.26 所示。这些输入为低电平有效,由小圆圈所指示。对于同步运算,这些预置位和清除输入必须保持在高电平。

图 7.27 给出了具有低电平有效的预置位($\overline{PRE}$)和清零($\overline{CLR}$)输入的边沿触发的 J-K 触发

器的逻辑图。这个图从根本上阐释了这些输入是怎样工作的。正如我们所看到的那样，这种连接使得它们无视同步输入 J、K 和时钟的影响。

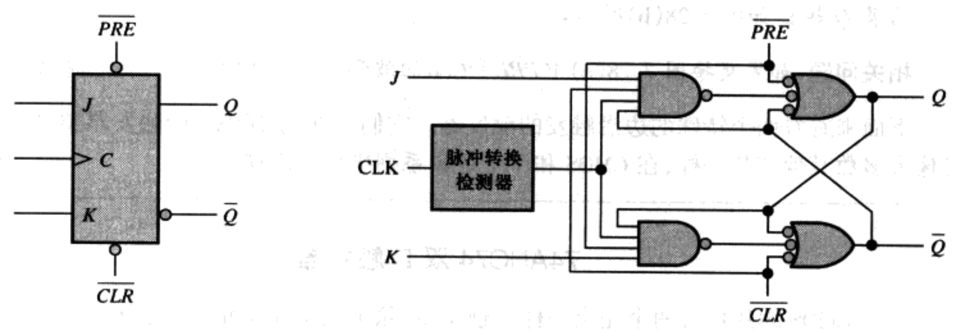


图 7.26　具有低电平有效的预置位和清零输入的J-K触发器的逻辑符号

图 7.27　具有低电平有效的预置位和清零输入的基本J-K触发器的逻辑图

例 7.8　对于图 7.28 中具有预置位和清零输入的上升沿 J-K 触发器，以及图 7.28(a)的时序图所示的输入，如果 Q 初始值为低电平，确定 Q 输出。

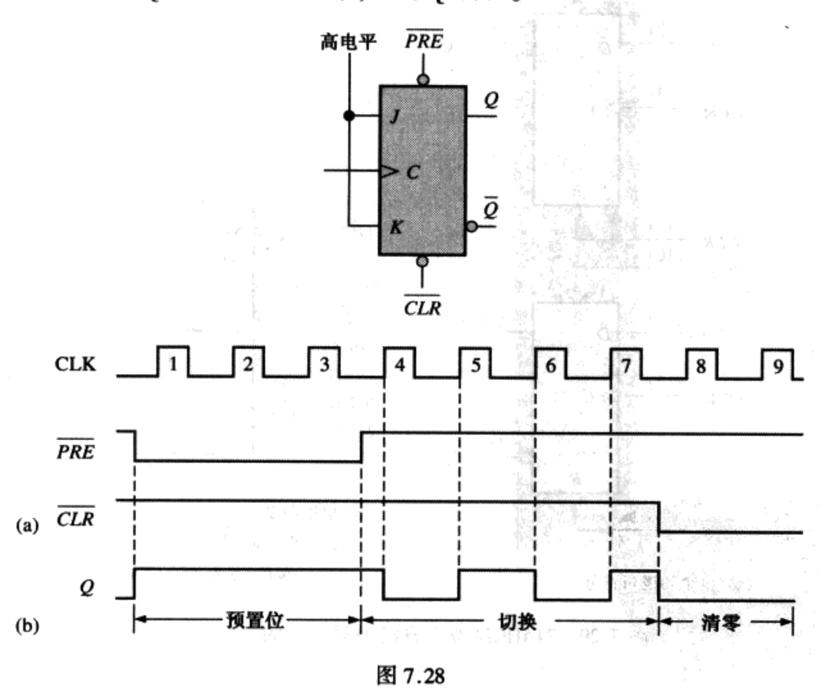


图 7.28

解：

1. 在时钟脉冲 1、2 和 3 期间，预置位($\overline{PRE}$)为低电平，无论同步输入 J 和 K 如何，触发器都保持为置位状态。
2. 对于时钟脉冲 4、5、6 和 7 来说，因为 J 为高电平，K 为高电平，并且 $\overline{PRE}$ 和 $\overline{CLR}$ 都是高电平，于是就会发生切换运算。

3. 对于时钟脉冲8和9来说，清零($\overline{CLR}$)输入为低电平，无论同步输入怎样，触发器都保持为复位状态。

结果 Q 输出如图7.28(b)所示。

相关问题：如果交换图7.28(a)中$\overline{PRE}$和$\overline{CLR}$的波形，那么 Q 输出应当是怎样的？

下面来看看两个特殊的边沿触发的触发器。它们是IC系列中不同触发器类型的代表，并且像大多数其他芯片一样，在CMOS和TTL逻辑系列中也会出现。

74AHC74 双 D 触发器

这个CMOS芯片包含两个完全一样的触发器，除了共享 V_{CC} 和接地之外都是相互独立的。这两个触发器都是上升沿触发，并且具有低电平有效异步预置位和清零输入。芯片中每个触发器的逻辑符号如图7.29(a)所示，而表示整个芯片的ANSI/IEEE标准单框图符号如图7.29(b)所示。引脚数目位于圆括号中。

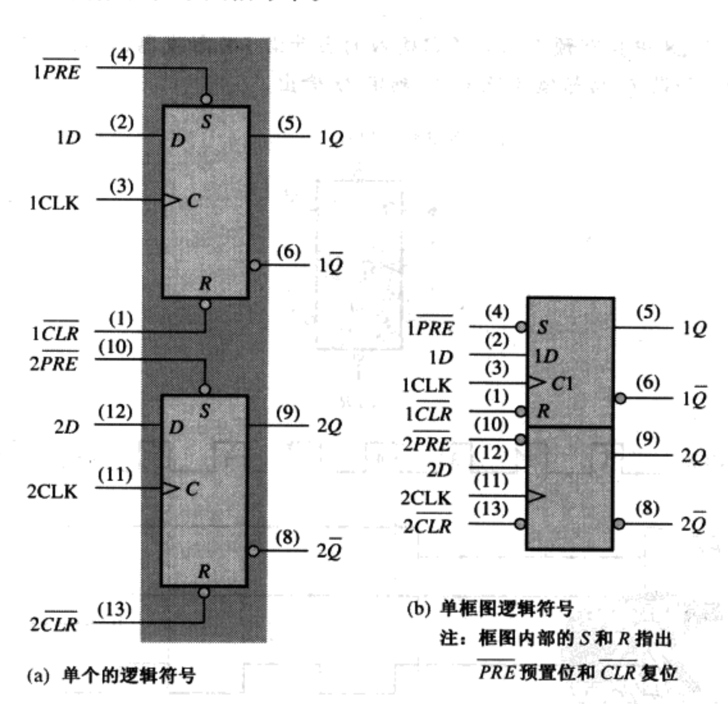


图7.29 74AHC74双上升沿D触发器的逻辑符号

74HC112 双 J-K 触发器

这种CMOS芯片也具有两个完全一样的触发器，它们是下降沿触发的，并且具有低电平有效异步预置位和清零输入。逻辑符号如图7.30所示。

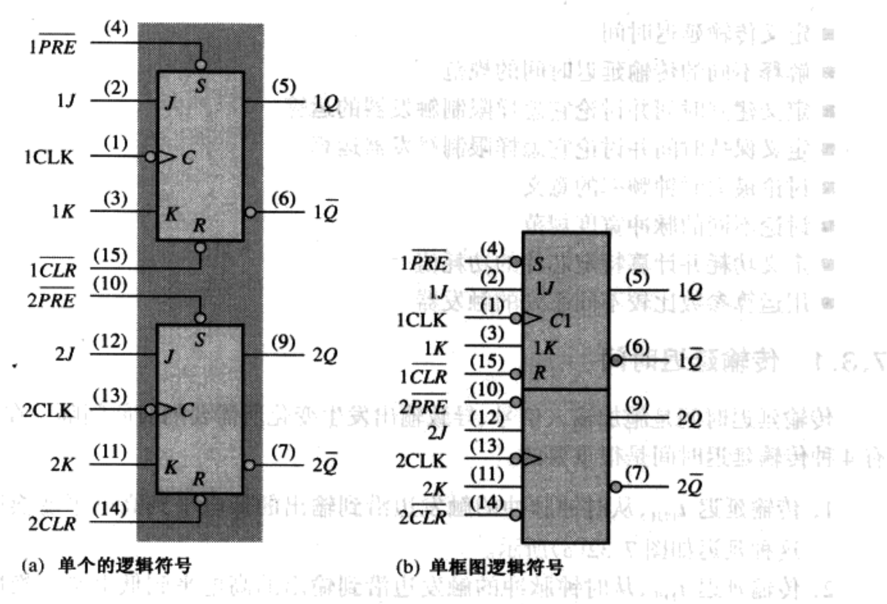


图 7.30　74HC112 双下降沿 J-K 触发器的逻辑符号

例 7.9　将图 7.31(a)中的 1*J*、1*K*、1CLK、1 $\overline{PRE}$和 1 $\overline{CLR}$波形加在 74HC112 芯片中的一个下降沿触发器上，确定 1*Q* 的输出波形。

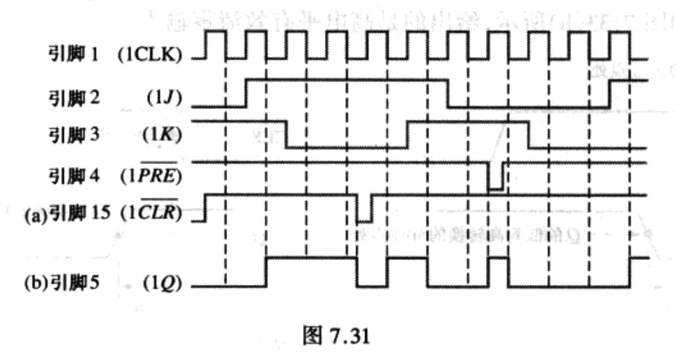


图 7.31

解：结果 1*Q* 波形如图 7.31(b)所示。注意每次低电平加在 1 $\overline{PRE}$或者 1 $\overline{CLR}$时，无论其他输入的状态如何，触发器就会置位或者复位。

相关问题：如果 1 $\overline{PRE}$和 1 $\overline{CLR}$波形进行交换，请确定 1*Q* 输出波形。

7.3　触发器运算特性

触发器的性能、运算需求及限制由芯片数据表上的几个运算特征或者参数来说明。一般来说，这些规范可以应用于所有的 CMOS 和 TTL 触发器。

学完本节以后，应当能够

- 定义传输延迟时间
- 解释不同的传输延迟时间的规范
- 定义建立时间并讨论它怎样限制触发器的运算
- 定义保持时间并讨论它怎样限制触发器运算
- 讨论最大时钟频率的意义
- 讨论不同的脉冲宽度规范
- 定义功耗并计算特定芯片的功耗值
- 用运算参数比较不同系列的触发器

7.3.1 传输延迟时间

传输延迟时间是施加输入信号、导致输出发生变化所需要的时间间隔。在触发器运算中，有4种传输延迟时间是很重要的：

1. 传输延迟 t_{PLH}，从时钟脉冲的触发边沿到输出的低电平到高电平变换所测得的时间。这种延迟如图7.32(a)所示。
2. 传输延迟 t_{PHL}，从时钟脉冲的触发边沿到输出的高电平到低电平变换所测得的时间。这种延迟如图7.32(b)所示。
3. 传输延迟 t_{PLH}，从预置位输入的前沿到输出低电平到高电平的变换之间测得的时间。这种延迟如图7.33(a)所示，给出的是低电平有效预置位输入。
4. 传输延迟 t_{PHL}，从清零输入的前沿到输出高电平到低电平的变换之间测得的时间。这种延迟如图7.33(b)所示，给出的是高电平有效清零输入。

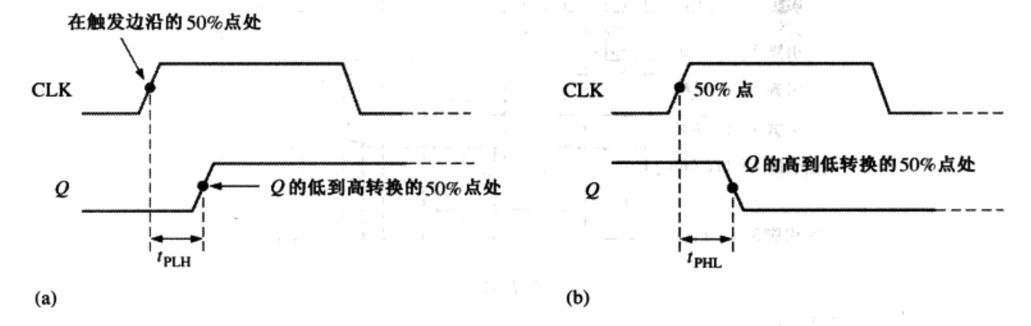


图7.32 传输延迟、时钟到输出

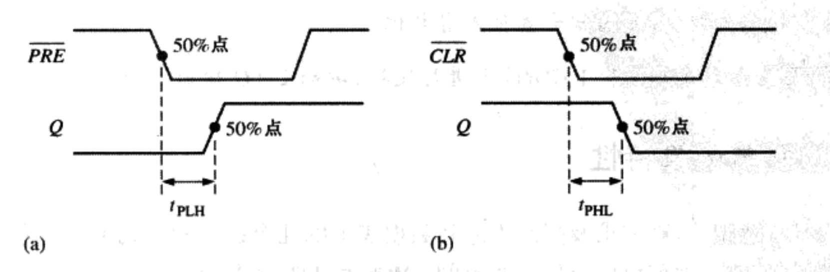


图7.33 传输延迟、预置位输入到输出和清零输入到输出

7.3.2 建立时间

建立时间(t_s)是输入先于时钟脉冲触发边沿到达所需要的最小时间间隔,在此时间里输入(J 和 K 或者 S 和 R,或者 D)的逻辑电平保持不变,这样就使得输入电平可靠地按时序进入触发器。这个时间间隔如图 7.34 中的 D 触发器所示。

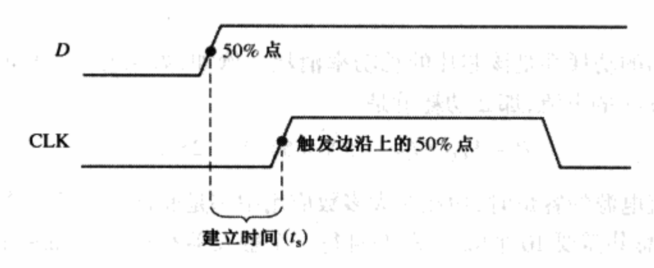


图 7.34 建立时间(t_s)。为了数据可靠进入,在时钟脉冲触发边沿到来之前,D输入上的逻辑电平必须出现的提前时间等于或者大于t_s

相应提示

CMOS 相比于 TTL 的一个优点是它可以运行在一个宽范围直流电源电压内(典型地为 2 V~6 V),因此可以使用没有精确调整的、不太昂贵的电源。同样,可以使用电池作为 CMOS 电路的第二或主要电源。此外,较低的电压就意味着 IC 具有较小的功率损耗。CMOS 的缺陷是它的性能在电源电压低的时候会差一些。例如,CMOS 触发器允许的最大时钟频率在 $V_{CC}=2$ V 时,这要比 $V_{CC}=6$ V 时低很多。

7.3.3 保持时间

保持时间(t_h)是在时钟脉冲触发边沿到达之后,输入上的逻辑电平需要保持的最小时间间隔,以使得输入电平可靠地按时序进入触发器。这个时间间隔如图 7.35 中的 D 触发器所示。

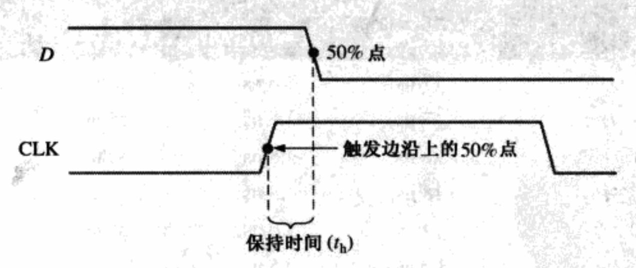


图 7.35 保持时间(t_h)。为了数据可靠进入,在时钟脉冲触发边沿到来之后,D输入上的逻辑电平必须保持的时间等于或者大于t_h

7.3.4 最大时钟频率

最大时钟频率(f_{max})是触发器能够可靠触发的最高速度。在最大值之上的时钟频率,触发器将不能足够快地做出响应,并且运算功能会减弱。

7.3.5 脉冲宽度

可靠运算的最小脉冲宽度(t_W)通常由生产商来为时钟、预置位和清零输入指定。典型的情况是,时钟由它的最小高电平时间和它的最小低电平时间来指定。

7.3.6 功耗

任何数字电路的功耗都是该芯片的总功率消耗。例如,如果在+5 V的直流电源上运行的触发器,需要5 mA的电流,那么功耗就是

$$P = V_{CC} \times I_{CC} = 5\text{ V} \times 5\text{ mA} = 25\text{ mW}$$

当考虑到直流电源的容量时,功耗在大多数应用中都是非常重要的。作为一个例子,假设有一个数字系统,总共需要10个触发器,并且每一个触发器都耗用25 mW的功率。总功率需求就是

$$P_T = 10 \times 25\text{ mW} = 250\text{ mW} = 0.25\text{ W}$$

这就给出了直流电源所需的输出容量。如果触发器运行在一个+5 V的直流电源上,那么电源必须提供的电流量为

$$I = \frac{250\text{ mW}}{5\text{ V}} = 50\text{ mA}$$

那么必须使用能够提供至少50 mA电流的+5 V直流电源。

7.3.7 触发器的具体比较

表7.5提供了4个同类型的CMOS和TTL触发器的比较,并依据本节所讨论的运行参数。

表7.5 同类型触发器的四个IC系列在25℃时的运算参数的比较

参数	CMOS		TTL	
	74HC74A	74AHC74	74LS74A	74F74
t_{PHL} (CLK到Q)	17 ns	4.6 ns	40 ns	6.8 ns
t_{PLH} (CLK到Q)	17 ns	4.6 ns	25 ns	8.0 ns
t_{PHL} ($\overline{CLR}$到Q)	18 ns	4.8 ns	40 ns	9.0 ns
t_{PLH} ($\overline{PRE}$到Q)	18 ns	4.8 ns	25 ns	6.1 ns
t_s (建立时间)	14 ns	5.0 ns	20 ns	2.0 ns
t_h (保持时间)	3.0 ns	0.5 ns	5 ns	1.0 ns
t_W (时钟高)	10 ns	5.0 ns	25 ns	4.0 ns
t_W (时钟低)	10 ns	5.0 ns	25 ns	5.0 ns
t_W($\overline{CLR}/\overline{PRE}$)	10 ns	5.0 ns	25 ns	4.0 ns
f_{max}	35 MHz	170 MHz	25 MHz	100 MHz
功率,静态	0.012 mW	1.1 mW		
功率,50%占空比			44 mW	88 mW

7.4　触发器应用

在本节中，将讨论触发器的三个常见应用，并讲解怎样使用它们。在第 8 章和第 9 章中，将详细介绍触发器在计数器和寄存器中的应用。

学完本节以后，应当能够

- 讨论数据存储中的触发器应用
- 描述触发器怎样应用于分频
- 解释在基本计数器应用中怎样使用触发器

7.4.1　并行数据存储

数字系统中的一个共同需求是在一组触发器中同时保存来自并行线的几个数据位。该运算如图 7.36(a)所示，图中使用了 4 个触发器。4 个并行数据线的每一个都连接到触发器的 D 输入上。触发器的时钟输入连接在了一起，使得每一个触发器都由相同的时钟脉冲来触发。在这个例子中，使用了上升沿触发器，所以在时钟的上升沿到来时，D 输入上的数据被触发器同时存储，如图 7.36(b)所示。同样，异步复位(R)输入连接到共同的$\overline{CLR}$线上，用来对所有的触发器进行初始复位。

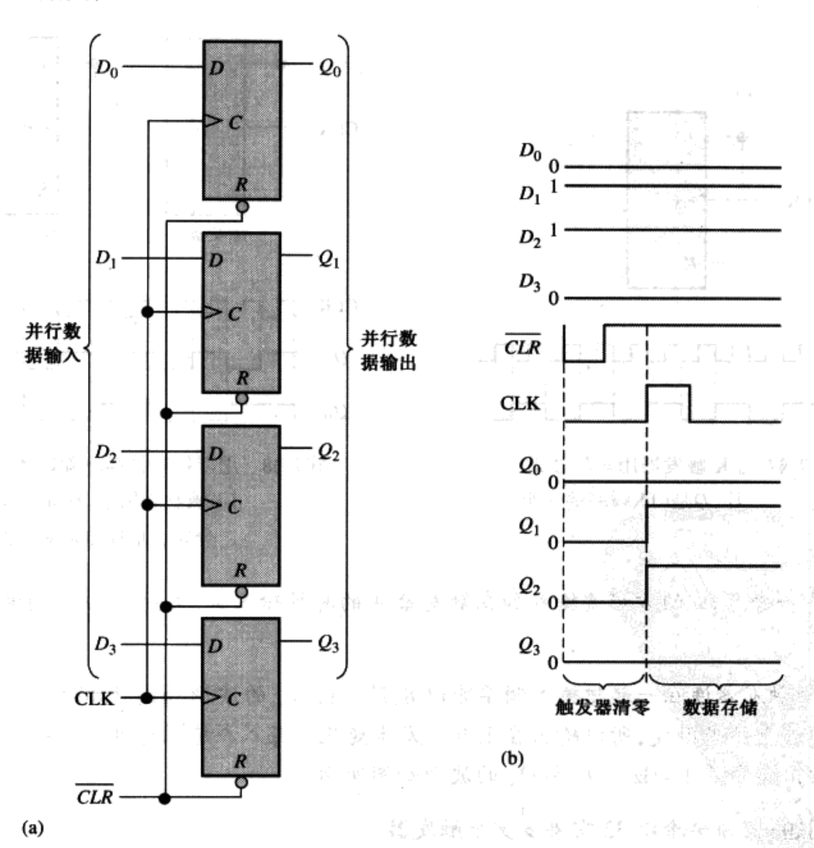


图 7.36　在基本寄存器中用来存储并行数据的触发器例子

这 4 个触发器组是用来存储数据的基本寄存器的一个例子。在数字系统中,数据一般以多位一组的形式存储(通常八位或者多位一组),位组可以表示数字、代码或者其他信息。寄存器将在第 9 章得到详细的介绍。

7.4.2 分频

触发器的另一个应用是对周期波形的频率进行分(减少)频。当脉冲波形加在一个 J-K 触发器的时钟输入时,J-K 触发器连接成切换状态($J = K = 1$),这时 Q 输出就是一个频率为时钟输入频率一半的方波。因此,单个触发器可以用做除 2 芯片,如图 7.37 所示。正如可以看到的那样,触发器在每一个触发时钟边沿(在这个例子中是上升沿触发)改变状态。这就产生了一个输出,它的频率变为时钟波形频率的一半。

时钟频率的进一步分频可以通过将触发器的输出用做第二个触发器的时钟输入来实现,如图 7.38 所示。Q_A 输出的频率由触发器 Q_B 二分频。所以 Q_B 输出频率就是原先时钟输入的 1/4。传输延迟时间没有在时序图中指出。

以这种方式连接触发器,就可以实现 2^n 分频,其中 n 是触发器的个数。例如,三个触发器把时钟频率除以 $2^3 = 8$;四个触发器把时钟频率除以 $2^4 = 16$;以此类推。

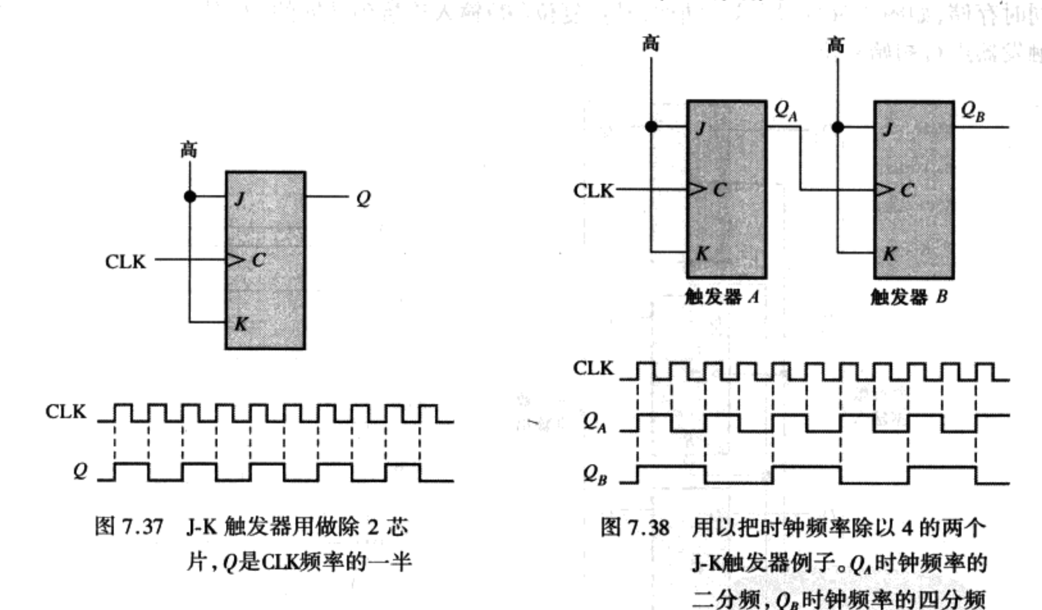


图 7.37 J-K 触发器用做除 2 芯片,Q是CLK频率的一半

图 7.38 用以把时钟频率除以 4 的两个J-K触发器例子。Q_A时钟频率的二分频,Q_B时钟频率的四分频

例 7.10 当一个 8 Hz 的方形波输入加在触发器 A 的时钟输入时,为图 7.39 中的电路开发 f_{out} 波形。

解:三个触发器连在一起把输入频率除以 8($2^3 = 8$),f_{out} 的波形如图 7.40 所示。因为这些触发器是上升沿触发,所以输出在上升沿发生变化。每八个输入脉冲就有一个输出脉冲,所以输出频率是 1 kHz。Q_A 和 Q_B 的波形如图所示。

相关问题:把频率除以 32 需要多少个触发器?

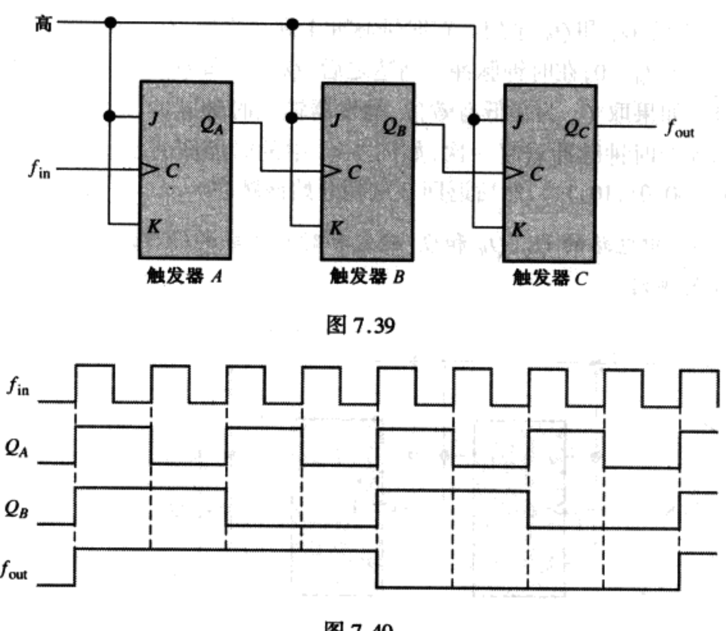


图 7.39

图 7.40

7.4.3 计数

触发器的另一个重要应用是在数字计数器方面,这将在第 8 章详细介绍。其概念如图 7.41 所示。这两个触发器是下降沿触发的 J-K 触发器。两个触发器的初始状态都是复位。触发器 A 在每个时钟脉冲的下降沿的转换处切换。触发器 A 的 Q 输出作为触发器 B 的时钟脉冲,所以每次 Q_A 从高电平到低电平的转换时,触发器 B 就会切换。Q_A 和 Q_B 波形的结果如图所示。

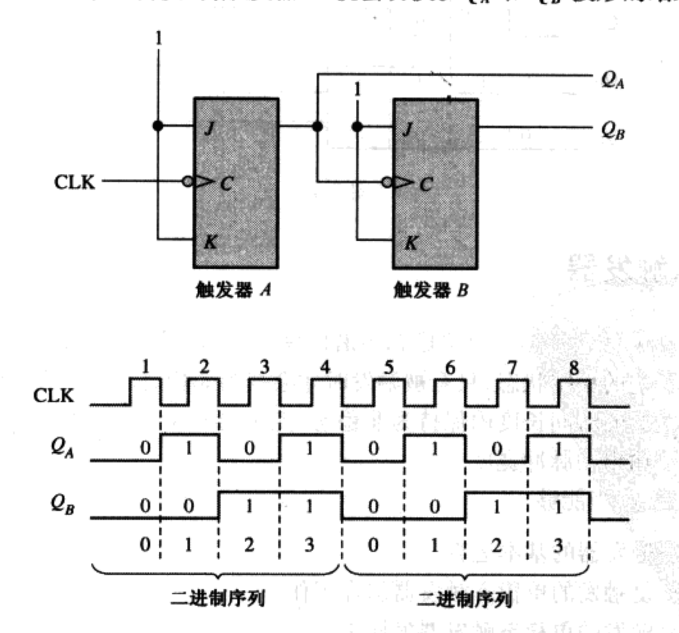


图 7.41　用来生成二进制计数序列的触发器。给出了两个(00,01,10,11) 循环

观察图 7.41 中的 Q_A 和 Q_B 序列。在时钟脉冲 1 到达之前，$Q_A=0$ 和 $Q_B=0$；在时钟脉冲 1 到达之后，$Q_A=1$ 和 $Q_B=0$；在时钟脉冲 2 到达之后，$Q_A=0$ 和 $Q_B=1$；在时钟脉冲 3 到达之后，$Q_A=1$ 和 $Q_B=1$。如果取 Q_A 为最低有效位、触发器输入时钟脉冲，就会产生 2 位的序列。这个二进制序列每 4 个时钟脉冲重复一次，如图 7.41 中的时序图所示。因此，这些触发器按顺序计数，从 0 到 3(00,01,10,11)，然后返回 0 重新开始该顺序。

例 7.11 为图 7.42 中电路的 Q_A、Q_B 和 Q_C 确定和时钟关联的输出波形，并给出由这些波形所表示的二进制序列。

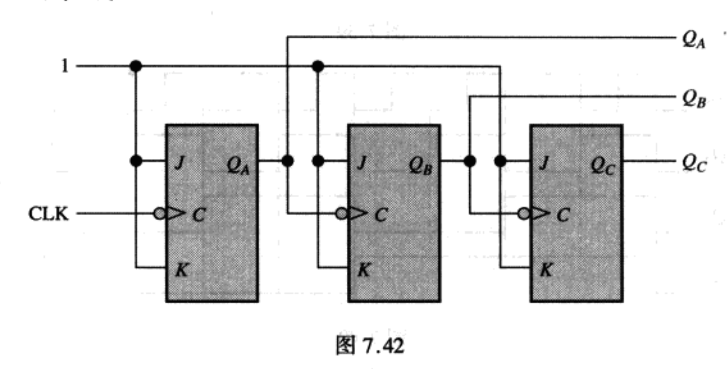


图 7.42

解：输出时序图如图 7.43 所示。注意输出在时钟脉冲的下降沿改变。输出经过的二进制序列 000,001,010,011,100,101,110,111 如图所示。

相关问题：产生表示十进制数 0 到 15 的二进制序列需要多少个触发器？

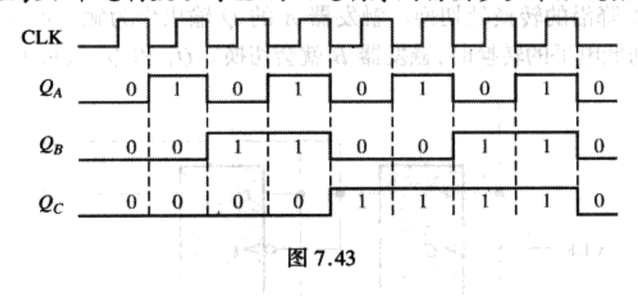


图 7.43

7.5 单稳态触发器

单稳态(one shot)触发器是一个单稳态多谐振荡器，就是只有一个稳定状态的芯片。单稳态触发器通常处于它的稳定状态，只有被触发时才会变为非稳态。一旦它被触发，单稳态触发器就会在事先确定好的时间长度内保持为非稳态，然后再自动回到稳定状态。芯片在非稳态所处的时间确定了输出的脉冲宽度。

学完本节以后，应当能够

- 描述单稳态触发器的基本运算
- 解释不可重复触发的单稳态触发器怎样工作
- 解释可重复触发的单稳态触发器怎样工作

■ 建立 74121 和 74LS122 单稳态触发器来得到指定的输出脉冲宽度
■ 识别施密特(Schmitt)触发符号并解释它的基本意义

◇ 单稳态触发器每触发一次就产生一个单脉冲。

图 7.44 给出了一个基本的单稳态触发器(单稳态多谐振荡器),它由一个逻辑门和一个反相器组成。当一个脉冲加到触发器的输入时,门 G_1 的输出就变为低电平。这个高电平到低电平的转换通过电容器耦合连接到反相器 G_2 的输入端。G_2 的视在低电平使得它的输出变为高电平。这个高电平往回连接到 G_1,使的它的输出保持低电平。此时触发器脉冲已经使得单稳态触发器的输出 Q 变为高电平。

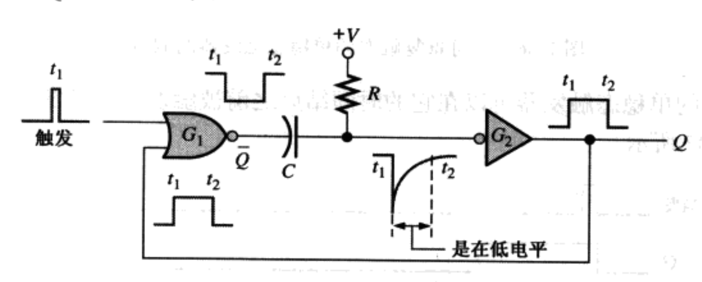


图 7.44　一个简单的单稳态触发器

电容立即开始通过电阻 R 充电直到高电平。充电的速率由 RC 时间常数来确定。当电容充电达到某个电平,并且这个电平对于 G_2 是一个高电平,这时输出就回到低电平。

为了进行总结,反相器 G_2 的输出响应触发器的输入变为高电平。它保持高电平的时间由 RC 时间常数设定。在这个时间结束时,输出变为低电平。所以单个窄触发脉冲产生单个输出脉冲,脉冲的持续时间由 RC 时间常数来控制。这个运算如图 7.44 所示。

典型的单稳态触发器的逻辑符号如图 7.45(a)所示,而具有外部电阻 R 和电容 C 的相同符号如图 7.45(b)所示。IC 单稳态触发器的两个基本类型是不可重复触发型和可重复触发型。

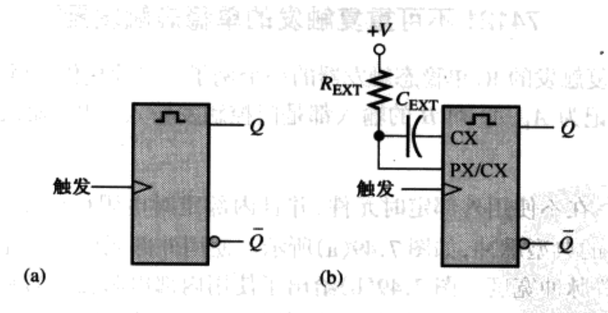


图 7.45　基本单稳态触发器逻辑符号,CX 和 RX 表示外部元件

不可重复触发的单稳态触发器从它触发进入非稳态到返回稳态这段时间,不会响应任何附加的触发脉冲。换句话说,在时间结束之前它将忽略任何触发脉冲。单稳态触发器保持为非稳态的时间是输出的脉冲宽度。

图 7.46 给出了不可重复触发的单稳态触发器,触发器分别在比它的脉冲宽度大的期间和比它的脉冲宽度小的期间被触发。注意在第二种情况下,附加脉冲被忽略了。

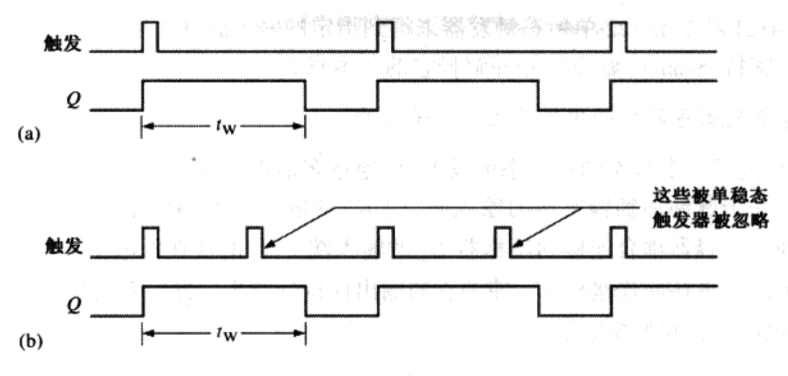


图 7.46 不可重复触发的单稳态触发器的响应

可重复触发的单稳态触发器可以在它的时间结束之前被触发。重触发的结果是脉冲宽度的延伸,如图 7.47 所示。

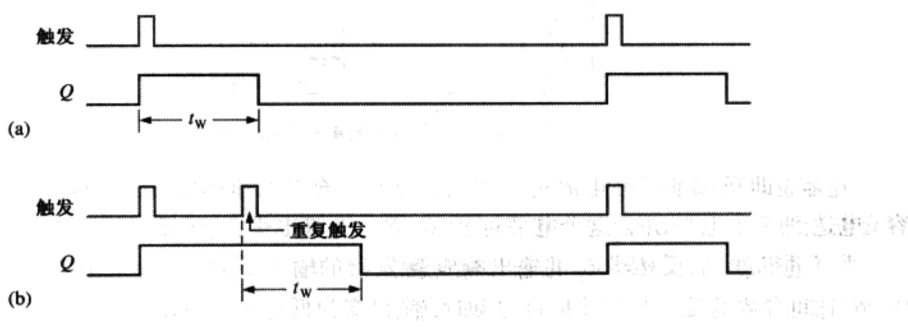


图 7.47 可重复触发的单稳态触发器的响应

74121 不可重复触发的单稳态触发器

74121 是不可重复触发的 IC 单稳态触发器的一个例子。芯片提供有外部电阻 R 和电容 C,如图 7.48 所示。标记为 A_1、A_2 和 B 的输入都是门控触发输入。R_{INT}输入连接到一个 2 kΩ 的内部定时电阻上。

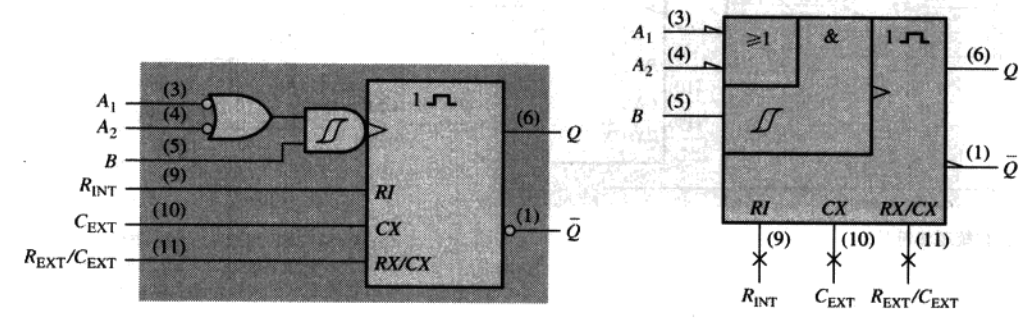
设置脉冲宽度 在不使用外部定时元件,并且内部定时电阻(R_{INT})连接到 V_{CC}时,就会产生一个宽度为 30 ns 的典型脉冲,如图 7.49(a)所示。使用外部元件可以在 30 ns 到 28 s 之间的任何时间间隔内设置脉冲宽度。图 7.49(b)给出了使用内部电阻(2 kΩ)和外部电容器的配置。图 7.49(c)给出了使用外部电阻和外部电容的配置。输出脉冲宽度由电阻值(R_{INT} = 2 kΩ 和选择 R_{EXT})及电容器来确定,公式如下所示:

$$t_W = 0.7RC_{EXT} \tag{7.1}$$

其中 R 是 R_{INT}或者 R_{EXT}。当 R 为千欧姆(kΩ)级和 C_{EXT}为皮法(pF)级时,输出脉冲宽度 t_W 就在纳秒(ns)级。

施密特触发符号 符号∫表示施密特输入。这种类型的输入使用产生磁滞现象的特殊

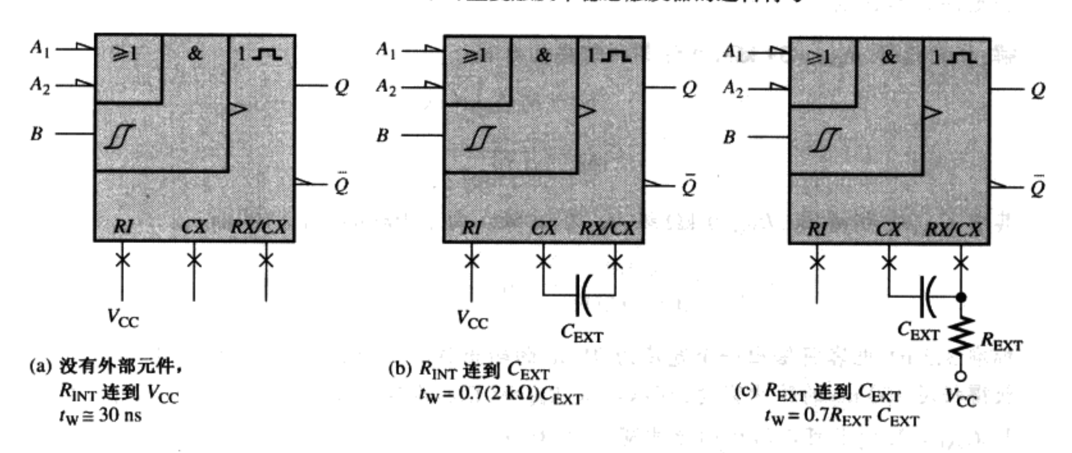
阈值电路,这种特征可以在缓慢变化的触发电压徘徊在临界输入电平周围时,防止状态之间的不稳定翻转。即使当输入的变化速度缓慢到 1 V/s 时,也可以产生可靠的触发。



(a) 传统逻辑符号

(b) ANSI/IEEE 标准 91-1984 逻辑符号(× = 非逻辑连接),"1⎍"是不可重复触发单稳态触发器的限定符号

图 7.48　74121 不可重复触发单稳态触发器的逻辑符号



(a) 没有外部元件,R_{INT} 连到 V_{CC}，$t_W \cong 30$ ns

(b) R_{INT} 连到 C_{EXT}，$t_W = 0.7(2\ \text{k}\Omega)C_{EXT}$

(c) R_{EXT} 连到 C_{EXT}，$t_W = 0.7R_{EXT}C_{EXT}$

图 7.49　设置 74121 的脉冲宽度的三种方法

74LS122 可重复触发单稳态触发器

74LS122 是具有清零输入的可重复触发 IC 单稳态触发器的一个例子。它也提供了外部电阻 R 和电容 C,如图 7.50 所示。标记为 A_1、A_2、B_1 和 B_2 的输入都是门控触发输入。

没有外部元件,可以得到大约为 45 ns 的最小脉冲宽度。使用外部元件可以得到更宽的脉冲宽度。对于特定的脉冲宽度(t_W)计算这些元件数值的一般公式如下:

$$t_W = 0.32RC_{EXT}\left(1 + \frac{0.7}{R}\right) \tag{7.2}$$

其中 0.32 是由特定类型的单稳态触发器确定的一个常数,R 为 kΩ 级,它可以是内部也可以是外部电阻,C_{EXT}为 pF 级,而 t_W 为 ns 级。内部电阻是 10 kΩ,可以用外部电阻替代[注意此式和 74121 的式(7.1)之间的区别]。

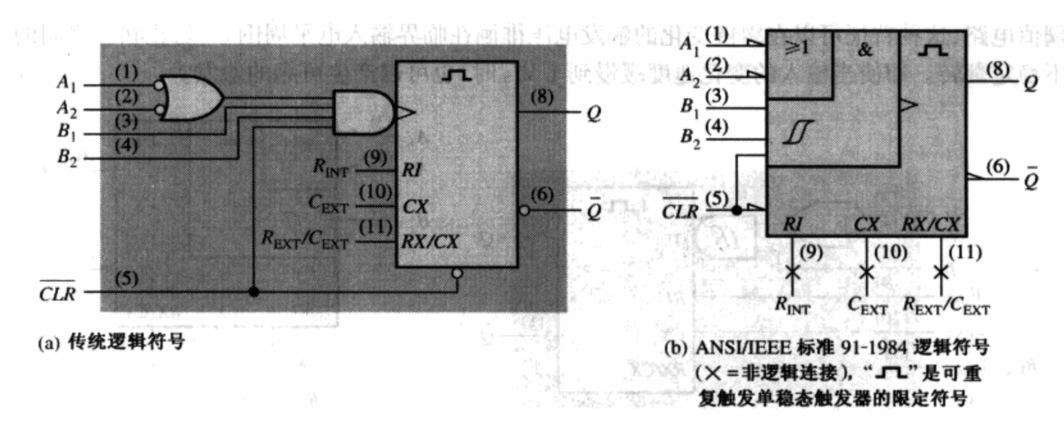


图 7.50 74LS122 可重触发单稳态触发器的逻辑符号

例 7.12 有一个应用需要一个单稳态触发器,脉冲宽度大约为 100 ms。使用 74121,给出连接和元件的参数值。

解:任意选择 $R_{EXT} = \mathbf{39\ k\Omega}$,并计算所需要的电容:

$$t_W = 0.7R_{EXT}C_{EXT}$$

$$C_{EXT} = \frac{t_W}{0.7R_{EXT}}$$

其中 C_{EXT}处为 pF 级,R_{EXT}为 kΩ 级,t_W 为 ns 级。由于 100 ms = 1×10^8 ns。

$$C_{EXT} = \frac{1\times10^8\ \text{ns}}{0.7(39\ \text{k}\Omega)} = 3.66\times10^{-6}\ \text{pF} = \mathbf{3.66\ \mu F}$$

标准 3.3 μF 电容将给出一个宽度为 91 ms 的输出脉冲,正确的连接如图 7.51 所示。为了获得接近 100 ms 的脉冲宽度,可以尝试 R_{EXT}和 C_{EXT}的其他数值组合。例如,R_{EXT} = 68 kΩ 和 C_{EXT} = 2.2 μF 可以给出的脉冲宽度为 105 ms。

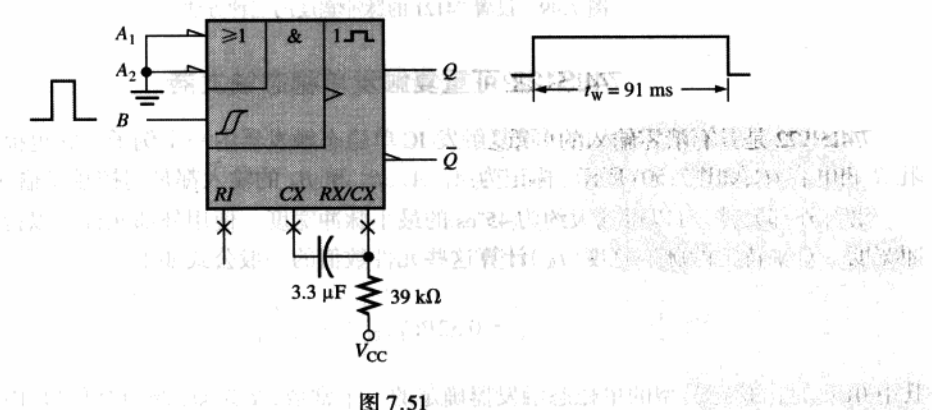


图 7.51

相关问题:使用外部电容并和 R_{INT}连接,用 74121 来产生宽度为 10 μs 的输出脉冲。

例 7.13　使用 74LS122 连接 R_{EXT}和 C_{EXT}，产生一个宽度为 1 μs 的输出脉冲，确定 R_{EXT}和 C_{EXT}的数值。

解：假设 C_{EXT} = **560 pF**，然后解出 R_{EXT}。脉冲宽度必须表示为 ns 级，C_{EXT}必须为 pF 级，R_{EXT}将是 kΩ 级。

$$
\begin{aligned}
t_W &= 0.32R_{EXT}C_{EXT}\left(1+\frac{0.7}{R_{EXT}}\right) = 0.32R_{EXT}C_{EXT} + 0.7\left(\frac{0.32R_{EXT}C_{EXT}}{R_{EXT}}\right) \\
&= 0.32R_{EXT}C_{EXT} + (0.7)(0.32)C_{EXT} \\
R_{EXT} &= \frac{t_W-(0.7)(0.32)C_{EXT}}{0.32C_{EXT}} = \frac{t_W}{0.32C_{EXT}} - 0.7 \\
&= \frac{1000\text{ ns}}{(0.32)560\text{ pF}} - 0.7 = \mathbf{4.88\ k\Omega}
\end{aligned}
$$

使用**4.7 kΩ**的标准数值。

相关问题：对于 74LS122 单稳态触发器，具有宽度为 5 μs 的输出脉冲，给出连接和元件的参数值。假设 C_{EXT} = 560 pF。

7.5.1　应用举例

一个实际的单稳态触发器应用是序列定时器，可以用来点亮一系列电灯。例如，这种类型的电路可以在高速公路建设项目中用于车道改变的定向指示器，或者用于车辆的系列转向信号。

图 7.52 给出了三个 74LS122 单稳态触发器连接在一起用做系列定时器。这个特殊电路产生 3 个 1 秒脉冲的序列。第一个单稳态触发器由开关关闭或者低频脉冲输入来触发，产生一个 1 秒输出脉冲。当第一个单稳态触发器(OS 1)定时结束时，1 秒脉冲就变为低电平，第二个单稳态触发器(OS 2)随即被触发，同样产生一个 1 秒输出脉冲。当第二个脉冲变为低电平时，第三个单稳态触发器(OS 3)就会被触发，并且产生第三个 1 秒脉冲。输出时序图如图所示。这种基本结构可以产生不同的定时输出。

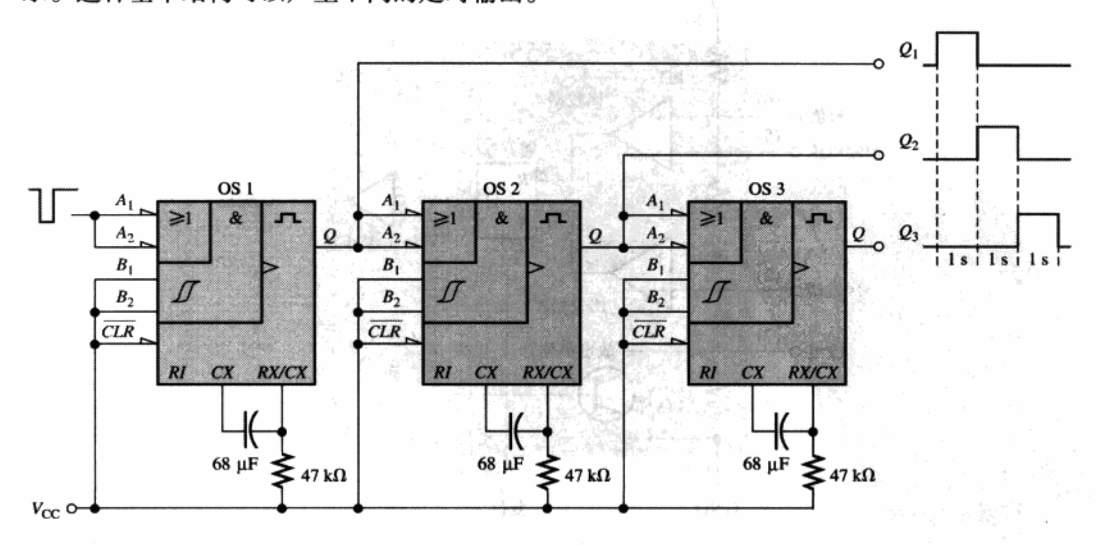


图 7.52　使用三个 74LS122 单稳态触发器的序列定时电路

7.6 555 定时器

555 定时器是功能多样和广泛应用的 IC 芯片，因为它可以用两种不同的模式来配置，要么是单稳态多谐振荡器(单稳态触发器)，要么是非稳态多谐振荡器(振荡器)。非稳态多谐振荡器没有稳定状态，因而在没有外部触发的条件下，会在两个非稳态之间来回变化(振荡)。

学完本节以后，应当能够

- 描述 555 定时器中的基本元件
- 把 555 定时器设置为单稳态触发器
- 把 555 定时器设置为振荡器

7.6.1 基本运算

◇ 555 定时器可以用做单稳态触发器(单稳)或者振荡器(非稳态)。

如图 7.53 所示给出 555 定时器内部元件的功能图。比较器芯片的特点是，当其正(+)输入上的电压大于负(-)输入上的电压时，输出就是高电平；而当负输入电压大于正输入电压时，输出就是低电平。分压器由三个 5 kΩ 的电阻组成，提供了 $1/3V_{CC}$的触发电平及 $2/3V_{CC}$的阈值电平。控制电压输入(引脚 5)可以根据需要，从外部把触发和阈值电平改变为其他的值。当通常情况下的高电平触发输入暂时低于 $1/3V_{CC}$时，比较器 B 的输出就从低电平变为高电平，并且使 S-R 锁存器置位，使得输出(引脚 3)变为高电平，并且使放电晶体管 Q_1 截止。输出将保持在高电平直至正常的低电平阈值输入高于 $2/3V_{CC}$，并且使得比较器 A 的输出从低电平变为高电平。这时锁存器复位，使得输出回到低电平并且使放电晶体管导通。外部的复位输入可以用来复位锁存器，此锁存器独立于阈值电路。触发和阈值输入(引脚 2 和引脚 6)由外部连接的元件来控制，以产生单稳态或者非稳态的效果。

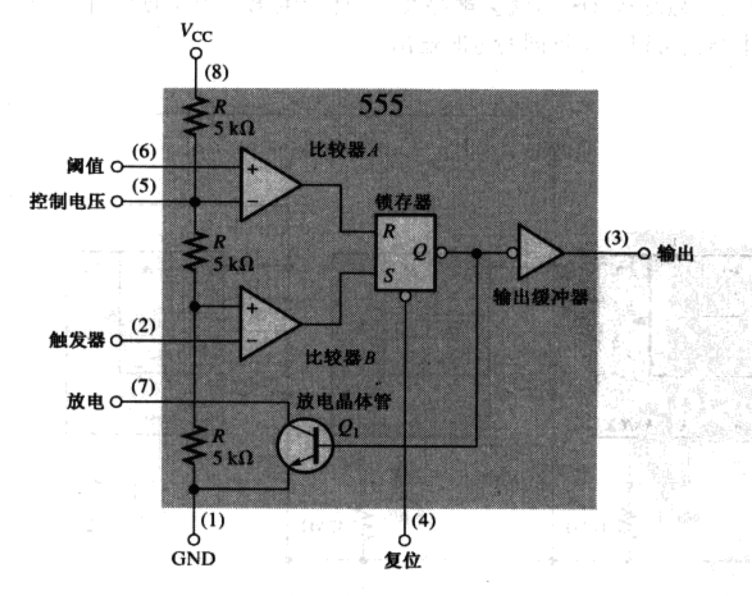


图 7.53 555 定时器(引脚编号在圆括号内)的内部功能图

7.6.2　单稳态(单稳态触发器)的运算

外部电阻和电容的连接如图 7.54 所示，用以设置 555 计时器为不可重复触发的单稳态触发器。输出的脉冲宽度由时间常数 R_1 和 C_1 根据下面的公式来确定：

$$t_W = 1.1R_1C_1 \tag{7.3}$$

没有使用控制电压输入，而是连接一个去耦电容器 C_2 来防止噪声影响触发和阈值电平。

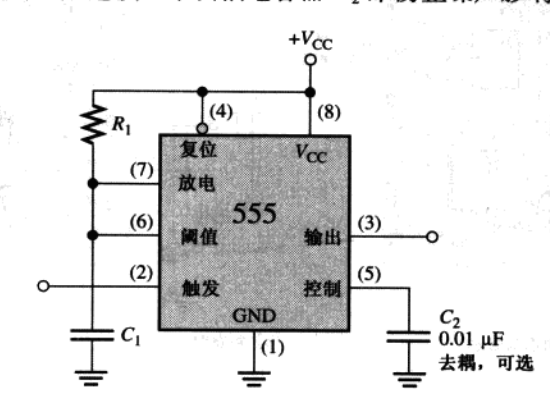


图 7.54　555 定时器连接成单稳态触发器

在施加触发脉冲之前，输出为低电平，放电晶体管 Q_1 导通，保持 C_1 放电，如图 7.55(a)所示。加上下降沿触发脉冲后，输出变为高电平而使得放电晶体管截止，允许电容器 C_1 开始通过 R_1 充电，如图 7.54(b)所示。当 C_1 充电到 1/3 V_{CC}时，输出将在 t_1 时间回到低电平，而 Q_1 立即导通，使 C_1 放电，如图 7.54(c)所示。正如所看到的那样，C_1 的充电速率决定了输出为高电平的时间长度。

例 7.14　$R_1 = 2.2\ \text{k}\Omega$、$C_1 = 0.01\ \mu\text{F}$ 的 555 单稳态电路的输出脉冲宽度是多少？

解：从式(7.3)得到脉冲宽度为

$$t_W = 1.1R_1C_1 = 1.1(2.2\ \text{k}\Omega)(0.01\ \mu\text{F}) = \mathbf{24.2\ \mu s}$$

相关问题：对于 $C_1 = 0.01\ \mu\text{F}$，确定脉冲宽度为 1 ms 的 R_1 的数值。

7.6.3　非稳态运算

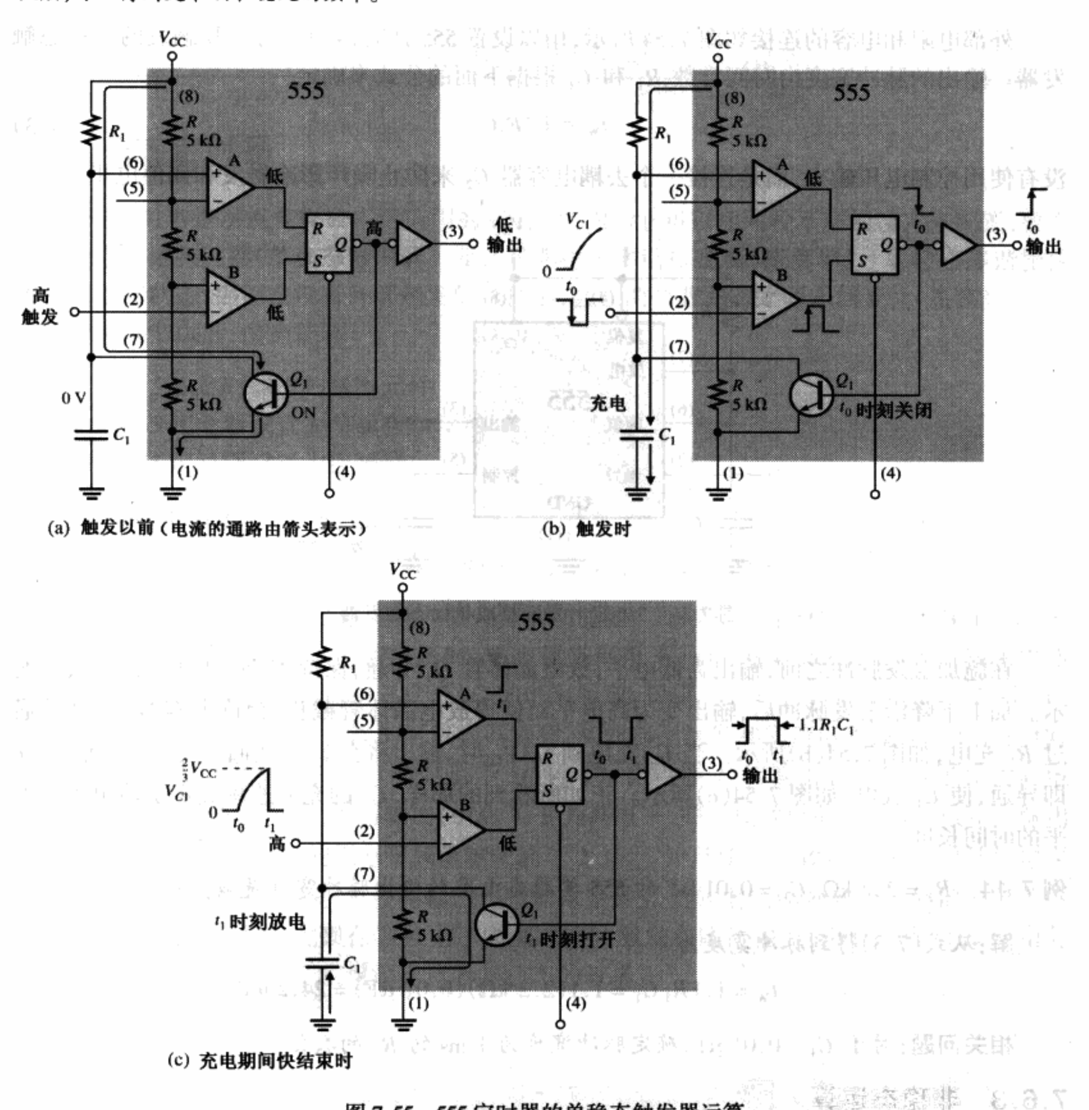
555 计时器连接后作为非稳态多谐振荡器使用，这是一种非正弦振荡器，如图 7.56 所示。注意阈值输入(*THRESH*)现在连接到了触发输入(*TRIG*)上。外部元件 R_1、R_2 和 C_1 构成了定时网络，用以设置振荡器的频率。0.01 μF 电容器 C_2 连接于控制(*CONT*)输入，C_2 严格用于去耦，对运算没有影响，在某些情况下可以去掉。



计算机小知识

所有的计算机都需要一个定时源来提供正确的时钟脉冲。定时部分控制所有的系统时序，并且负责系统

硬件的正确运算。定时部分通常由晶体控制的振荡器和用以分频的计数器组成。使用高频振荡器并分频为低频,可以得到更准确和稳定的频率。



(a) 触发以前(电流的通路由箭头表示)

(b) 触发时

(c) 充电期间快结束时

图 7.55 555 定时器的单稳态触发器运算

开始时,当电源打开后,电容器(C_1)并没有充电,因此触发电压(引脚 2)是 0 V。这导致了比较器 B 的输出为高电平,比较器 A 的输出为低电平,迫使锁存器的输出,也就是基极 Q_1 为低电平,并保持晶体管截止。这时,C_1 开始通过 R_1 和 R_2 充电,如图 7.57 所示。当电容器电压达到 1/3 V_{CC}时,比较器 B 就变为它的低电平输出状态;而当电容器电压达到 2/3 V_{CC}时,比较器 A 就变为它的高电平输出状态。这就使锁存器复位,使得基极 Q_1 变为高电平,并使晶体管导通。这一连串变化为电容器建立了一个从 R_2 到晶体管的放电回路,如图所示。电容器就在循环开始时放电,使得比较器 A 变为低电平。当电容器放电到 1/3 V_{CC}时,比较器 B 就变换

为高电平；这使得锁存器置位，并使得基极 Q_1 变为低电平，并且关闭晶体管。然后开始另一个充电循环，重复整个过程。结果就是一个矩形波输出，它的占空比取决于电阻 R_1 和 R_2 的数值。振荡的频率由下面的公式给出，或者可以使用图 7.58 中的图表找到：

$$f = \frac{1.44}{(R_1 + 2R_2)C_1} \tag{7.4}$$

通过选择 R_1 和 R_2，可以调节输出的占空比。由于 C_1 通过 $R_1 + R_2$ 充电并仅通过 R_2 放电，因此如果 $R_2 \gg R_1$，就可以达到最接近 50% 的占空比，从而使得充电和放电时间大致相等。

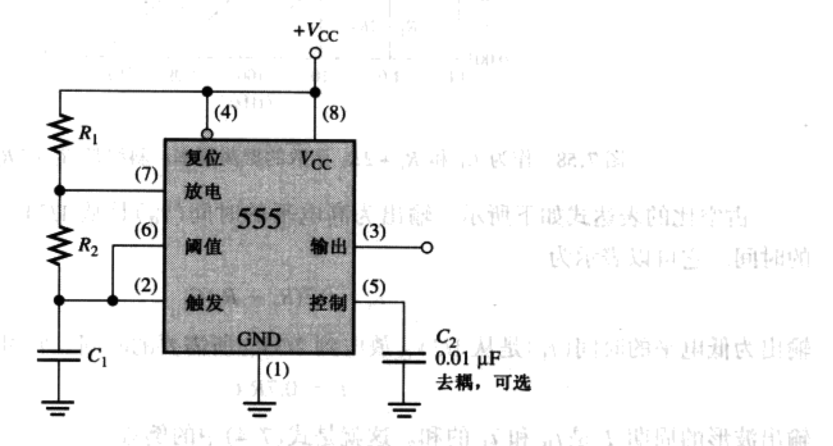


图 7.56　连接为非稳态多谐振荡器（振荡器）的 555 定时器

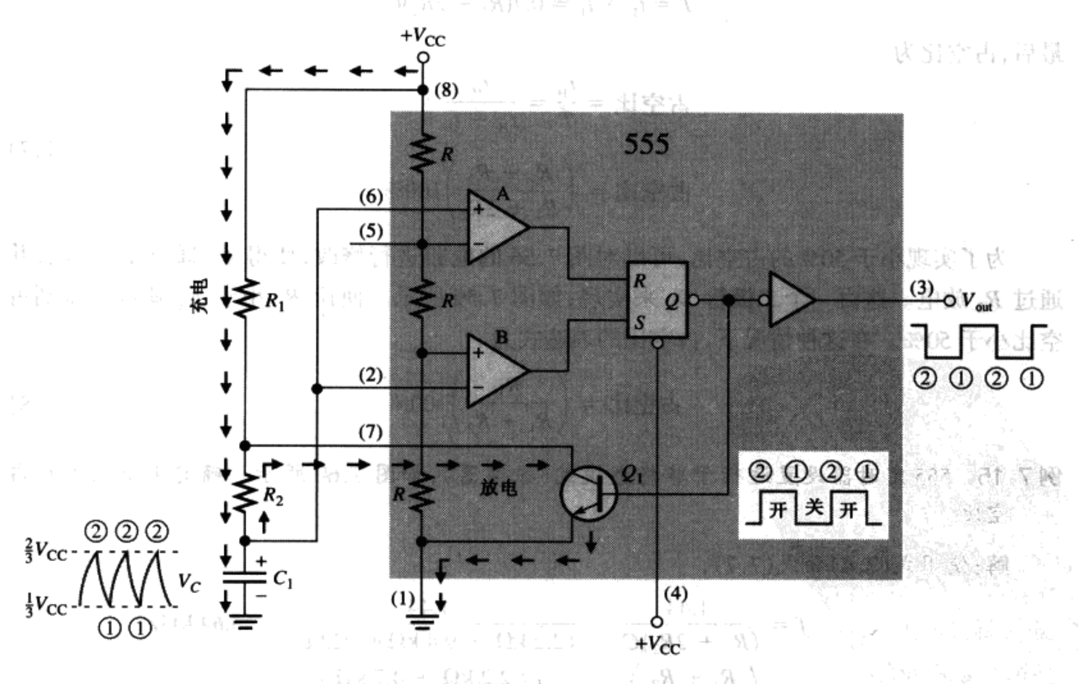


图 7.57　在非稳态模式下的 555 定时器的运行

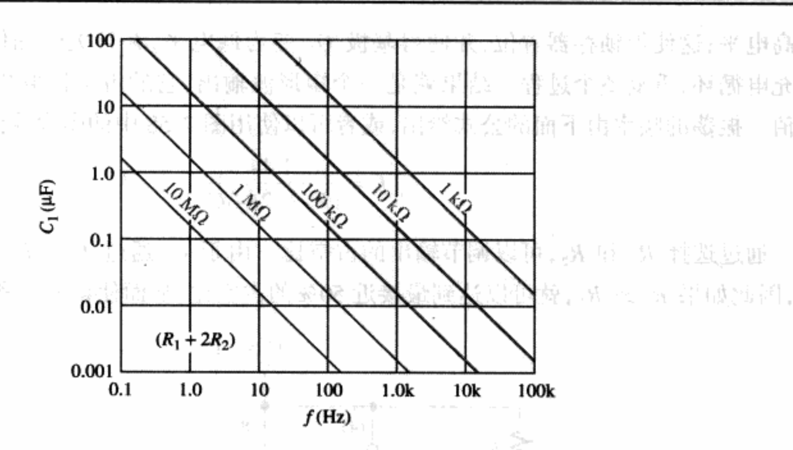


图 7.58 作为 C_1 和 R_1+2R_2 函数的振荡频率。斜线是 R_1+2R_2 的值

占空比的表达式如下所示。输出为高电平的时间(t_H)是从 $1/3V_{CC}$ 充电到 $2/3V_{CC}$ 所需要的时间。它可以表示为

$$t_H = 0.7(R_1 + R_2)C_1 \tag{7.5}$$

输出为低电平的时间(t_L)是从 $1/3V_{CC}$ 放电到 $2/3V_{CC}$ 所需要的时间。它可以表示为

$$t_L = 0.7R_2C_1 \tag{7.6}$$

输出波形的周期 T 是 t_H 和 t_L 的和。这就是式(7.4)中的倒数。

$$T = t_H + t_L = 0.7(R_1 + 2R_2)C_1$$

最后,占空比为

$$\text{占空比} = \frac{t_H}{T} = \frac{t_H}{t_H + t_L}$$
$$\text{占空比} = \left(\frac{R_1 + R_2}{R_1 + 2R_2}\right)100\% \tag{7.7}$$

为了实现小于50%的占空比,可以对图 7.56 的电路进行修改,使得 C_1 通过 R_1 充电,并通过 R_2 放电。这由一个二极管 D_1 来实现,如图 7.59 所示。使让 R_1 小于 R_2 就可以做到占空比小于50%。在这种情况下,占空比的表达式为

$$\text{占空比} = \left(\frac{R_1}{R_1 + R_2}\right)100\% \tag{7.8}$$

例 7.15 555 定时器设置运行于非稳态模式(振荡器),如图 7.60 所示。确定输出频率和占空比。

解:使用式(7.4)和式(7.7),

$$f = \frac{1.44}{(R_1 + 2R_2)C_1} = \frac{1.44}{(2.2\,\text{k}\Omega + 9.4\,\text{k}\Omega)0.022\,\mu\text{F}} = \mathbf{5.64\,kHz}$$

$$\text{占空比} = \left(\frac{R_1 + R_2}{R_1 + 2R_2}\right)100\% = \left(\frac{2.2\,\text{k}\Omega + 4.7\,\text{k}\Omega}{2.2\,\text{k}\Omega + 9.4\,\text{k}\Omega}\right)100\% = \mathbf{59.5\%}$$

相关问题: 如图7.59所示,在 R_2 两端跨接一个二极管,确定图7.60中的占空比。

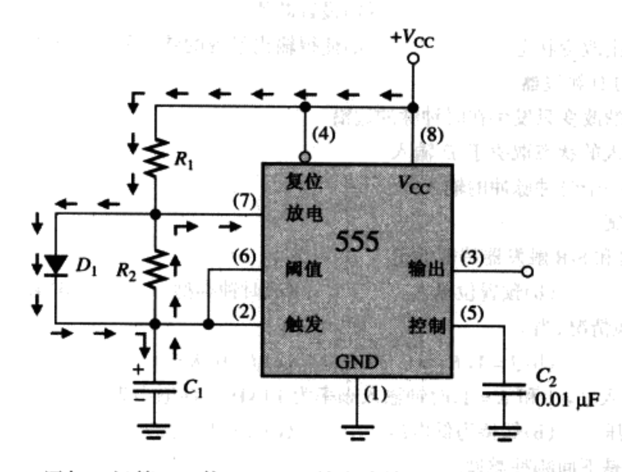


图7.59 添加二极管 D_1,使 $R_1 < R_2$,就允许输出的占空比调节到小于50%

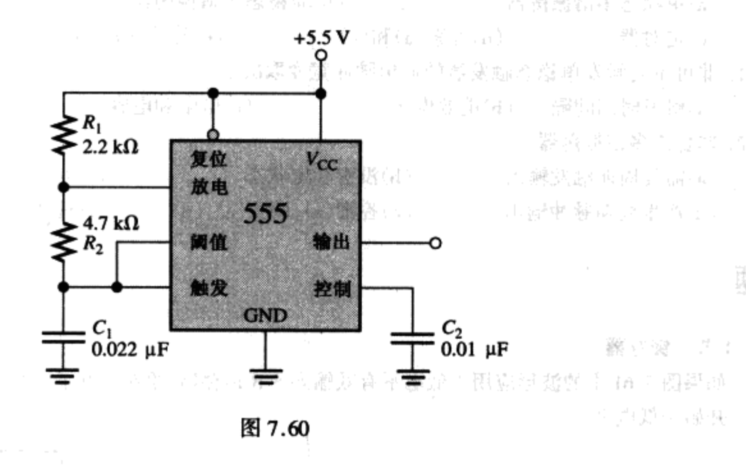


图7.60

自测题 (答案在本章结尾)

1. 如果S-R锁存器在 S 输入有一个1,在 R 输入上有一个0,然后 S 输入变为0,那么锁存器将会________。
   (a)置位 (b)复位 (c)无效 (d)清零
2. 当出现下列哪种条件时,S-R锁存器就会出现无效状态?
   (a)$S=1,R=0$ (b)$S=0,R=1$ (c)$S=1,R=1$ (d)$S=0,R=0$
3. 对于门控D锁存器,哪种情况下 Q 输出总是等于 D 输入?
   (a)在启动脉冲之前 (b)在启动脉冲期间
   (c)在启动脉冲之后 (d)答案(b)和(c)
4. 跟锁存器相似,触发器所属的逻辑电路类别称为
   (a)单稳态多谐振荡器 (b)双稳态多谐振荡器
   (c)非稳态多谐振荡器 (d)单次振荡器

5. 触发器的时钟输入的目的是
(a)清除芯片 (b)设置芯片
(c)总是使得输出改变状态 (d)使得输出呈现的状态取决于控制(S-R、J-K 或者 D)输入
6. 对于边沿触发的 D 触发器
(a)触发器状态的改变只发生在时钟脉冲边沿
(b)触发器要进入的状态取决于 D 输入
(c)输出跟随每一个时钟脉冲的输入
(d)所有这些答案
7. 区分 J-K 触发器和 S-R 触发器特征的是
(a)切换情况 (b)预置位输入 (c)时钟类型 (d)清零输入
8. 触发器处于切换情况,当
(a) $J=1, K=0$ (b) $J=1, K=1$ (c) $J=0, K=0$ (d) $J=0, K=1$
9. J-K 触发器的输入 $J=1$ 和 $K=1$,时钟输入频率为 10 kHz。Q 输出为
(a)保持为高电压 (b)保持为低电压 (c) 10 kHz 方波 (d)5 kHz 方波
10. 单稳态触发器是下面哪种类型?
(a)单稳态多谐振荡器 (b)非稳态多谐振荡器
(c)定时器 (d)答案(a)和(c) (e)答案(b)和(c)
11. 非可重复触发单稳态触发器的输出脉冲宽度取决于
(a)触发时间间隔 (b)电源电压 (c)电阻和电容 (d)阈值电压
12. 非稳态多谐振荡器
(a)需要周期触发输入 (b)没有稳定状态 (c)是一个振荡器
(d)产生周期脉冲输出 (e)答案(a),(b),(c),(d) (f)只是答案(b),(c),(d)

习题

7.1 节 锁存器

1. 如果图 7.61 中的波形应用于低电平有效输入 S-R 锁存器,绘制出和输入相关的结果 Q 输出。假设 Q 开始于低电平。

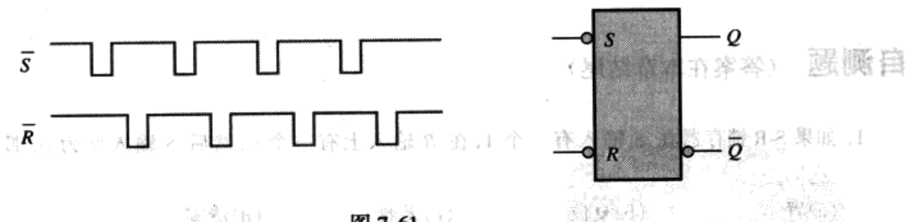


图 7.61

2. 在图 7.62 中的输入波形加到低电平有效的 S-R 锁存器上,给出如习题 1 的解。

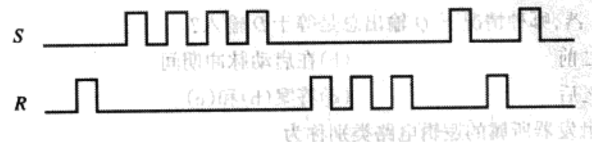


图 7.62

3. 对与图 7.63 的输入波形,给出习题 1 的解。

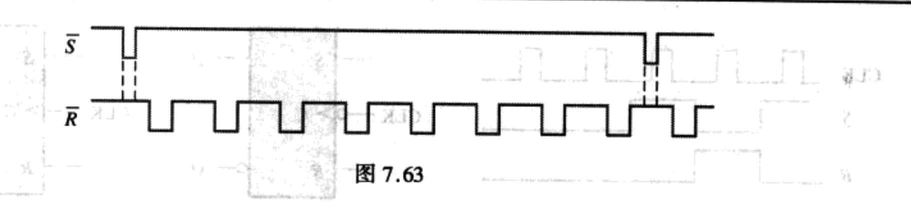


图 7.63

4. 门控 S-R 锁存器的输入如图 7.64，确定输出 Q 和 $\overline{Q}$。给出它们和使能输入的正确关系。假设 Q 开始时是低电平。

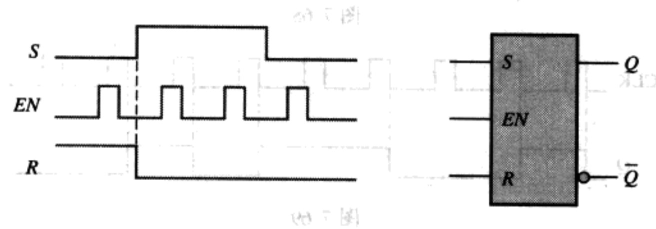


图 7.64

5. 对于图 7.65 的输入，求习题 4 的解。

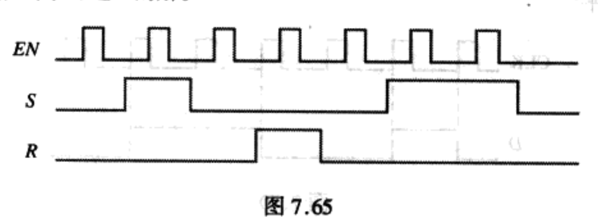


图 7.65

6. 对于图 7.66 的输入，求习题 4 的解。

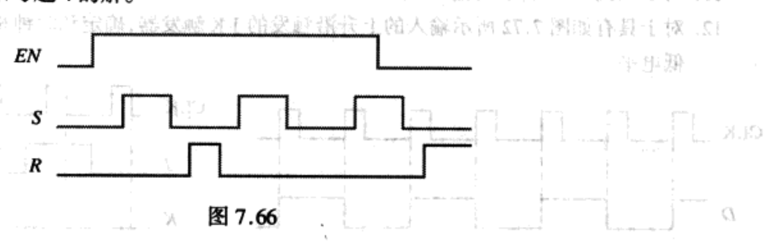


图 7.66

7. 观察到的门控 D 锁存器的输入波形如图 7.67 所示。如果锁存器的初始状态为复位，绘制时序图，指出所期望看到的输出 Q 的波形。

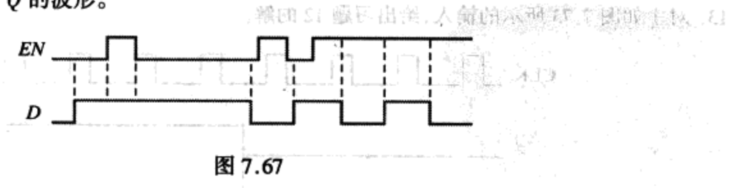


图 7.67

7.2 节　边沿触发的触发器

8. 两个边沿触发的 S-R 触发器如图 7.68 所示。如果输入如图所示的那样，绘制出相和时钟关联的每个触发器的 Q 输出，并解释两者之间的区别。触发器初始状态为复位。
9. 边沿触发的 S-R 触发器和时钟信号关联的 Q 输出如图 7.69 所示。确定产生这个输出所需要的 S 和 R 输入上的输入波形，假设触发器为上升沿触发类型。

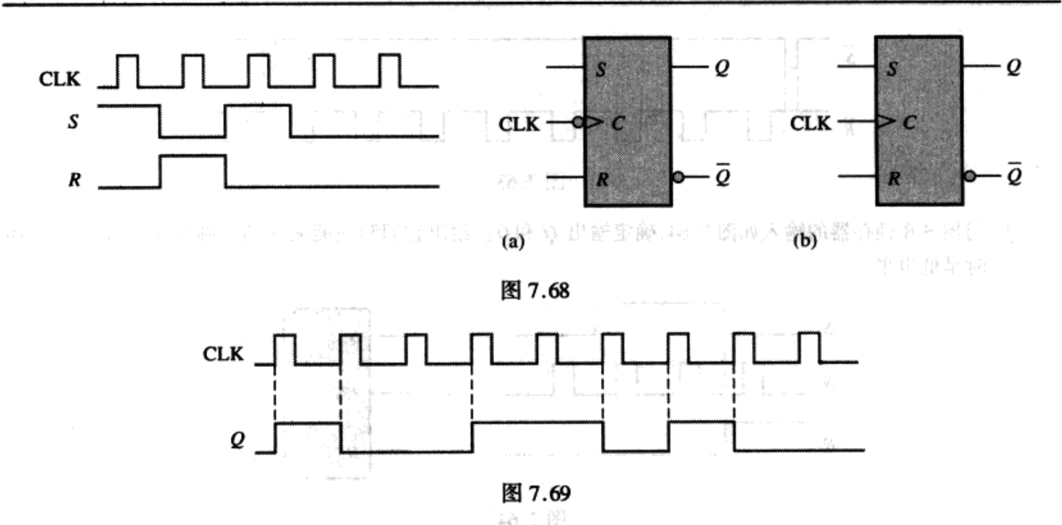


图 7.68

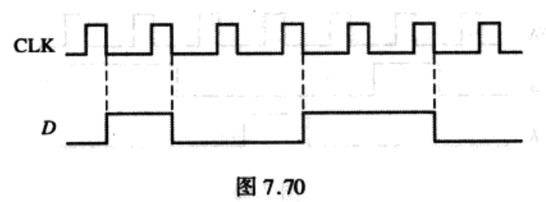


图 7.69

10. 对于具有如图 7.70 所示输入的 D 触发器,绘制出和时钟相关联的 Q 输出。假设为上升沿触发并且 Q 初始为低电平。

图 7.70

11. 对于如图 7.71 所示的输入,给出习题 10 的解。
12. 对于具有如图 7.72 所示输入的上升沿触发的 J-K 触发器,确定和时钟相关联的输出。假设 Q 开始于低电平。

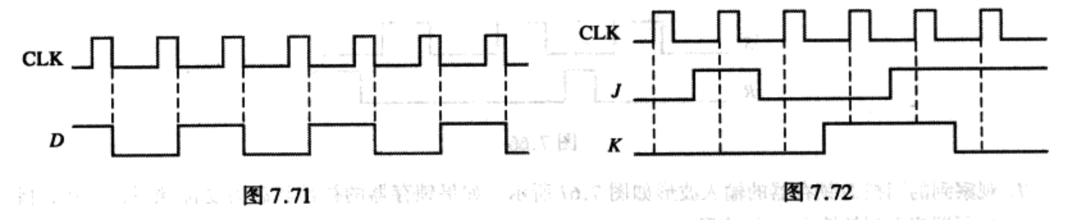


图 7.71 图 7.72

13. 对于如图 7.73 所示的输入,给出习题 12 的解。

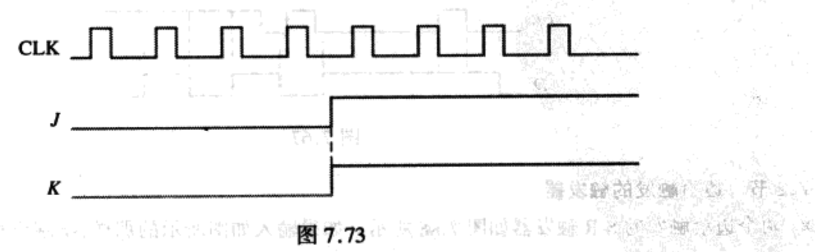


图 7.73

14. 如果如图 7.74 所示的信号加在 J-K 触发器的输入,确定和时钟相关联的 Q 波形。假设 Q 初始为低电平。

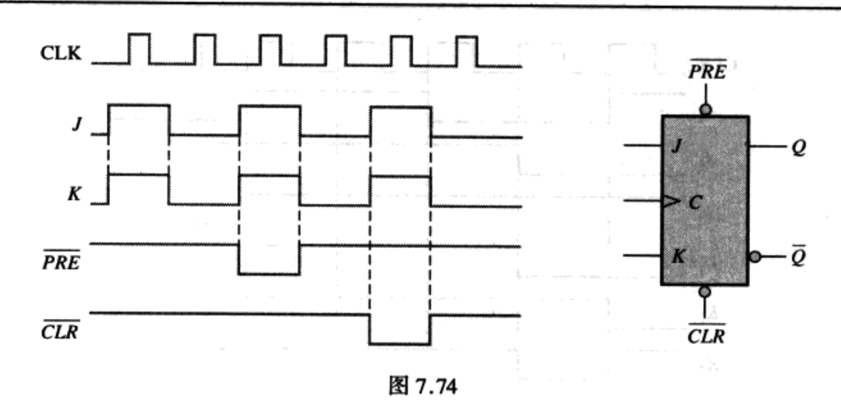


图 7.74

15. 对于具有如图 7.75 所示输入的下降沿触发的 J-K 触发器，给出和时钟相关联的 Q 输出。假设 Q 初始为低电平。

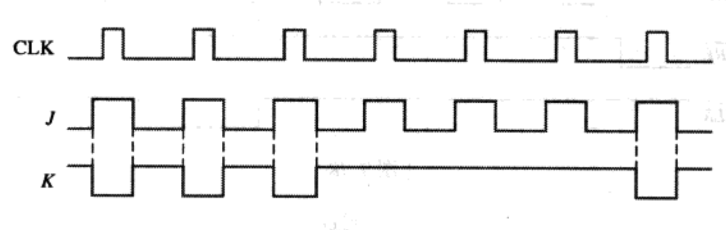


图 7.75

16. 如图 7.76 所示，下面的串行数据通过与门加在触发器上。确定出现在输出 Q 上所得到的串行数据。每个位时间都有一个时钟脉冲。假设 Q 初始为 0，并且 $\overline{PRE}$ 和 $\overline{CLR}$ 都是高电平。最右边的位首先加入。

J_1：1010011

J_2：0111010

J_3：1111000

K_1：0001110

K_2：1101100

K_3：1010101

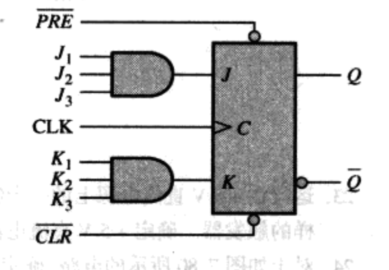


图 7.76

17. 为图 7.76 的电路在图 7.77 中完成时序图，画出输出 Q（初始为低电平）。假设 $\overline{PRE}$ 和 $\overline{CLR}$ 保持为高电平。

18. 以相同的 J 和 K 输入，但是 $\overline{PRE}$ 和 $\overline{CLR}$ 和时钟的关系如图 7.78所示，解出习题 17 的问题。

7.3 节　触发器运算特性

19. 触发器的功率损耗由什么决定？

20. 一般情况下，生产商的数据表指定和触发器相关的 4 种不同的传输延迟时间。给出每一种时间的名称并加以描述。

21. 某个触发器的数据表指定时钟脉冲的最小高电平时间为 30 ns，最小低电压时间为 37 ns。那么最大运行频率是多少？

22. 如图 7.79 所示的触发器初始状态为复位。给出输出 Q 和时钟脉冲之间的关系，假设传输延迟 t_{PLH}（时钟脉冲到 Q）是 8 ns。

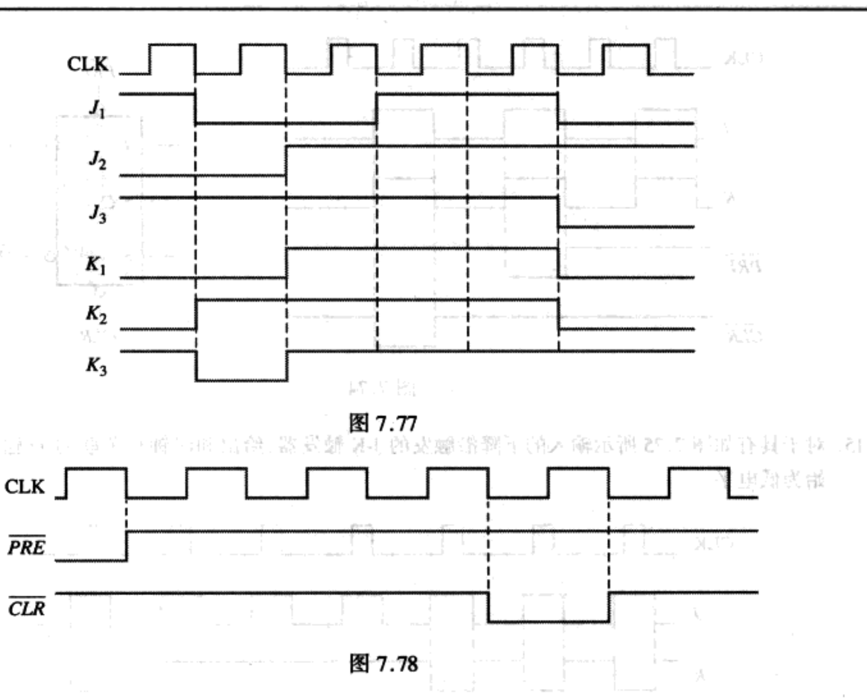


图 7.77

图 7.78

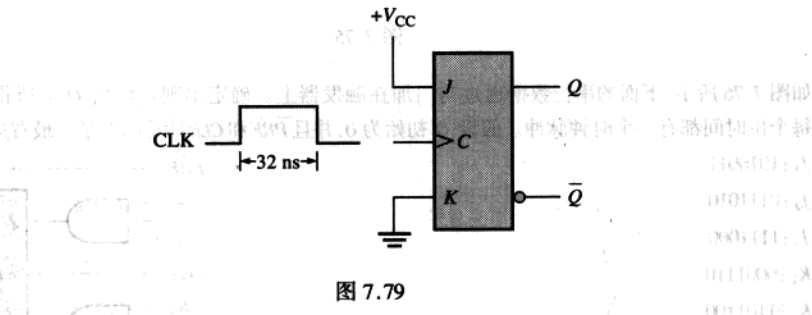


图 7.79

23. 运行在 + 5 V 直流电源上的一个特殊触发器所需要的直流电流为 10 mA。某个数字设备使用 15 个这样的触发器。确定 + 5 V 直流电源所需要的电流容量及该系统的总功耗。
24. 对于如图 7.80 所示的电路，确定可靠运行下的最大时钟信号频率，假设每个触发器的建立时间为 2 ns，并且从时钟到输出的延迟时间(t_{PLH} 和 t_{PHL})为 5 ns。

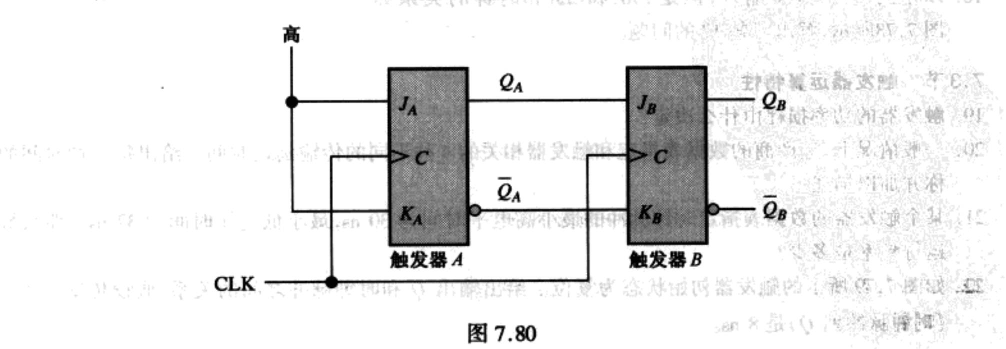


图 7.80

7.4 节　触发器应用

25. 某个 D 触发器的连接方式如图 7.81 所示。确定和时钟关联的输出 Q。这个设备有什么样的特殊功能？

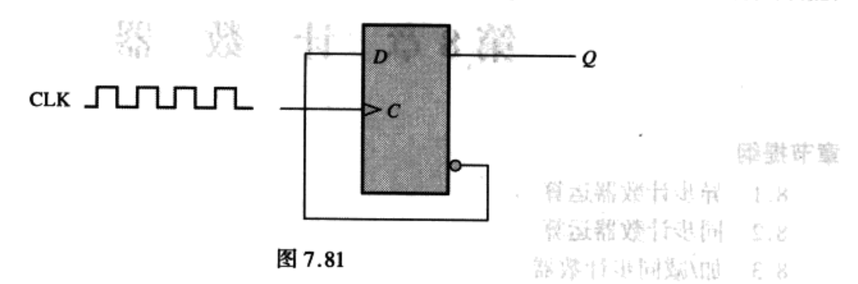


图 7.81

26. 对于图 7.80 中的电路，为 8 个时钟脉冲开发一个时序图，给出与时钟关联的输出 Q_A 和 Q_B。

7.5 节　单稳态触发器

27. 如果外部电阻为 3.3 kΩ 和外部电容为 2000 pF，确定 74121 单稳态触发器的脉冲宽度。
28. 74LS122 单稳态触发器要产生一个 5 μs 时间间隔的输出脉冲。使用 10 000 pF 的电容，确定所需要的外部电阻的值。

7.6 节　555 定时器

29. 使用 555 定时器组成单稳态触发器，产生一个 0.25 s 的输出脉冲。
30. 对 555 定时器进行配置，作为非稳态多谐振荡器运行，如图 7.82 所示。确定它的频率。
31. 把 555 定时器作为一个非稳态多谐振荡器使用，输出频率为 20 kHz。如果外部电容 C 为 0.002 μF，占空比大约是 75%，确定外部电阻的值。

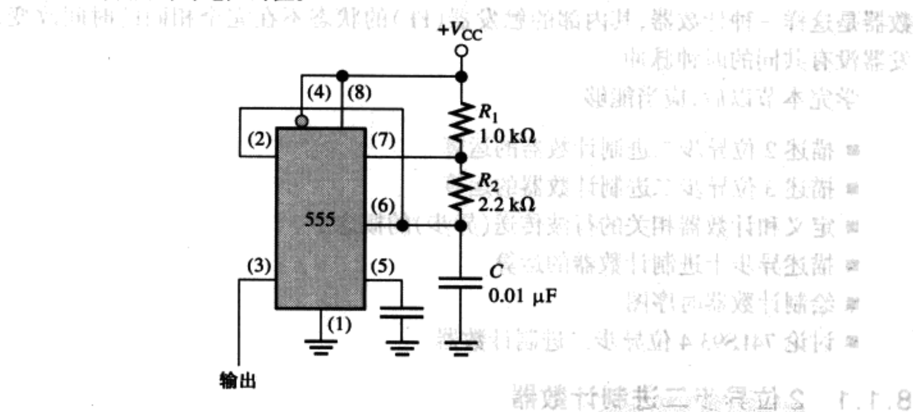


图 7.82

自测题答案

1. (a)　2. (c)　3. (d)　4. (b)　5. (d)　6. (d)　7. (a)　8. (b)　9. (d)　10. (d)
11. (c)　12. (f)

第8章　计　数　器

章节提纲

8.1　异步计数器运算

术语异步是指事件相互之间没有固定的时间关系，并且一般来说不在同时发生。异步计数器是这样一种计数器，其内部的触发器(FF)的状态不在完全相同的时间改变，因为这些触发器没有共同的时钟脉冲。

学完本节以后，应当能够

- 描述2位异步二进制计数器的运算
- 描述3位异步二进制计数器的运算
- 定义和计数器相关的行波传送(异步)的概念
- 描述异步十进制计数器的运算
- 绘制计数器时序图
- 讨论74LS93 4位异步二进制计数器

8.1.1　2位异步二进制计数器

图8.1给出了一个连接成异步运算的2位计数器。注意时钟(CLK)只应用于第一个触发器(FF0)的时钟输入(C)上，FF0总是处在最低有效位(LSB)。第二个触发器(FF1)由FF0的输出来触发。FF0在每个时钟脉冲的上升沿改变状态，但是FF1状态的改变仅发生在FF0输出的$\overline{Q}_0$上升沿转换的时间。由于通过触发器的固有的传输延迟，因此输入时钟脉冲(CLK)的转换和FF0输出$\overline{Q}_0$的转换绝对不可能发生在同一时间。所以，这两个触发器永远不会同时被触发，因而该计数器的运算是异步的。

◇ 异步计数器的时钟输入总是只连接到最低有效位(LSB)的触发器上。

时序图　通过加入四个时钟脉冲到FF0，观察每个触发器的输出，检查一下图8.1中异步

计数器的基本运算。图 8.2 解释了对应于时钟脉冲的触发器输出状态的变化。两个触发器连接为切换运算($J=1,K=1$),并且假设它们初始状态为复位(Q 为低电平)。

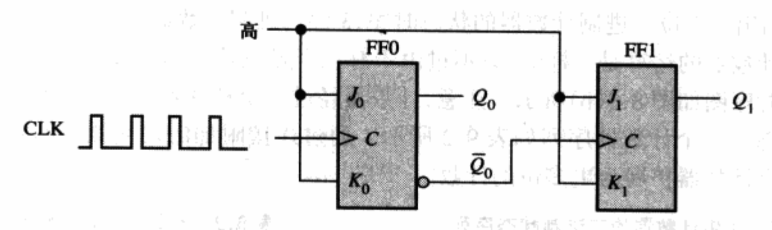


图 8.1 2 位异步二进制计数器

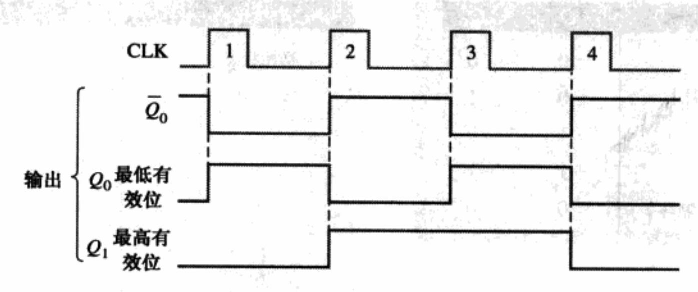


图 8.2 图 8.1 中计数器的时序图

◇ *异步计数器也称为行波(前级的输出连接后级的输入,信号的传送就像波的行进)计数器。*

CLK1(时钟脉冲 1)的上升沿使得 FF0 的 Q_0 输出变为高电平,如图 8.2 所示。与此同时,输出$\overline{Q_0}$变为低电平,但是它对 FF1 没有影响,因为必须发生上升沿的转换才使触发器触发。在 CLK1 的上升沿之后,$Q_0=1$ 和 $Q_1=0$。CLK2 的上升沿使得 Q_0 变为低电平。输出 $\overline{Q}_0$ 变为高电平,而触发器 FF1 使得 Q_1 变为高电平。在 CLK2 的上升沿之后,$Q_0=0$ 和 $Q_1=1$。CLK3 的上升沿再次使得 Q_0 变为高电平。$\overline{Q}_0$ 输出变为低电平,并且对 FF1 没有影响。因此,在 CLK3 的上升沿之后,$Q_0=1$ 和 $Q_1=1$。CLK4 的上升沿使得 Q_0 变为低电平,同时 $\overline{Q}_0$ 变为高电平,而触发器 FF1 使得 Q_1 变为低电平。在 CLK4 的上升沿之后,$Q_0=0$ 和 $Q_1=0$。现在计数器已经再次循环到了它的原始状态(两个触发器都是复位状态)。

在时序图中,Q_0 和 Q_1 输出的波形如图 8.2 所示,它们和时钟脉冲关联。为了简化的目的,Q_0、Q_1 和时钟脉冲的转换以同步形式给出,尽管这是个异步计数器。当然,在 CLK 和 Q_0 转换之间及在 $\overline{Q}_0$ 和 Q_0 的转换之间,有一定的延迟。

注意在图 8.2 中,2 位计数器呈现了 4 种不同的状态,正如对于两个触发器($2^2=4$)所期望的那样。当然,如果 Q_0 表示最低有效位(LSB)而 Q_1 表示最高有效位(MSB),计数器状态的顺序表示为表 8.1 所列出的一系列二进制数。

◇ *在数字逻辑中,Q_0 总是最低有效位(LSB),除非特别说明。*

由于计数器经历了一个二进制序列,所以图 8.1 中的计数器是一个二进制计数器。它实际上计数了 3 个时钟脉冲,而在第 4 个脉冲上,再循环到它的原始状态($Q_0=0,Q_1=0$)。术语再循环一般用在计数器运算上,它是指计数器从它的最终状态返回到原始状态的转换。

8.1.2　3 位异步二进制计数器

表 8.2 列出了 3 位二进制计数器的状态时序,3 位二进制计数器如图 8.3(a)所示。该基本运算和二位计数器的运算是一样的,只不过由于有三个触发器,3 位计数器有八个状态。八个时钟脉冲的时序图如图 8.3(b)所示。注意,计数器经过一个从 0 到 7 的二进制计数,然后再循环回到 0 状态。二个计数器序列如表 9.2 所示。通过连接附加的切换状态的触发器,可以很容易地将这个计数器扩展为更多位的计数。

表 8.1　图 8.1 中计数器的二进制状态序列

时钟输入	Q_1	Q_0
初始状态	0	0
1	0	1
2	1	0
3	1	1
4（再循环）	0	0

表 8.2　3 位二进制状态序列

时钟输入	Q_2	Q_1	Q_0
初始状态	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8（再循环）	0	0	0

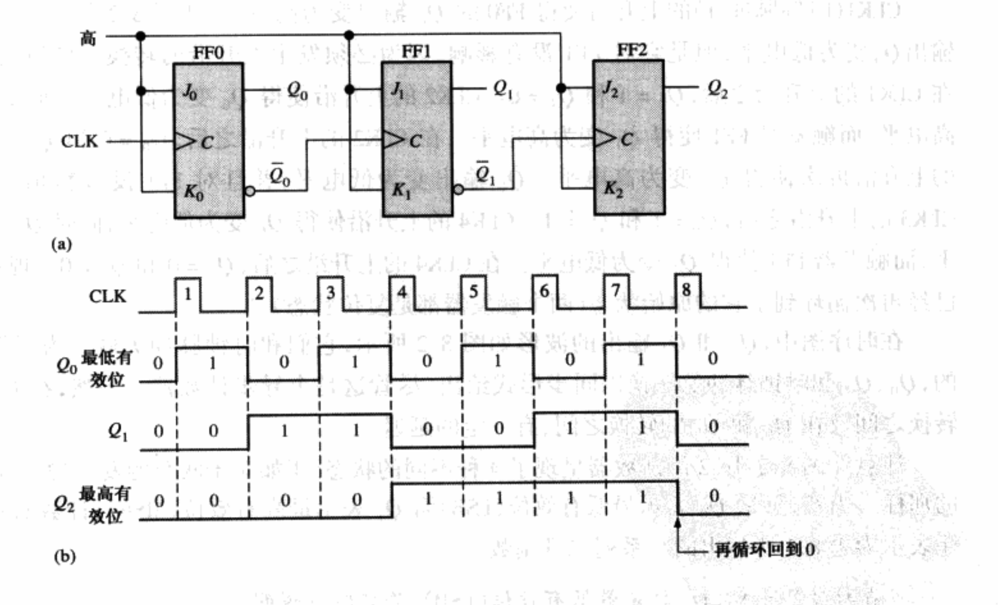


图 8.3　3 位异步二进制计数器和一个周期的时序图

传输延迟　由于下述原因,异步计数器常常称为行波(ripple)计数器:输入时钟脉冲的效应首先被 FF0“感觉”到。由于通过 FF0 的传输延迟,这个效应不能立即到达 FF1。然后,在 FF2 被

触发之前，还有一个通过 FF1 的传输延迟。因此，输入时钟脉冲的效应以“行波”的形式通过计数器，由于传输延迟，花费了一些时间后才能使输入效应到达最后一个触发器。

为了说明这一点，注意图 8.3 中计数器的所有三个触发器都在 CLK4 的上升沿改变状态。这种行波时钟效应如图 8.4 所示，其中给出了前 4 个时钟脉冲，同时指出了传输延迟。Q_0 从高电平到低电平的转换在时钟脉冲的上升沿转换后的一个延迟时间(t_{PLH})内发生。Q_1 从高电平到低电平的转换在时钟脉冲 $\bar{Q}_0$ 的上升沿转换之后的一个延迟时间(t_{PLH})内发生。Q_2 从高电平到低电平的转换在时钟脉冲 $\bar{Q}_1$ 的上升沿转换之后的一个延迟时间(t_{PLH})内发生。正如我们所看到的那样，FF2 直到时钟脉冲 CLK4 上升沿之后的两个延迟后才被触发。因此，时钟脉冲 CLK4 的效应要花费三个传输延迟时间，并以“行波”的形式通过计数器，然后才能使 Q_2 从低电平变为高电平。

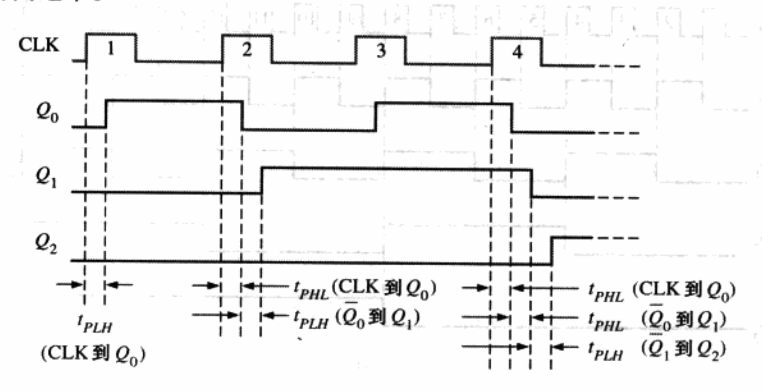


图 8.4 3 位异步(行波时钟)二进制计数器的传输延迟

异步计数器的这种积累延迟在许多应用中都是一个主要的缺点，因为它限制了计数器按时钟运行的速度，并且产生了译码问题。计数器中的最大积累延迟必须小于时钟波形的周期。

例 8.1 4 位异步二进制计数器如图 8.5(a)所示。每个触发器都是下降沿触发，并且具有传输延迟 10 ns。画出一个时序图，给出每个触发器的输出 Q，并且确定从时钟脉冲的触发边沿到 Q_3 状态发生相应变化之间的总传输延迟时间。同时确定该计数器可以运行的最大时钟频率。

解： 忽略延迟的时序图如图 8.5(b)所示。对于总延迟时间，CLK8 或者 CLK16 的效应必须在 Q_3 改变之前通过 4 个触发器，所以

$$t_{p(\text{总时间})} = 4 \times 10\ \text{ns} = 40\ \text{ns}$$

最大时钟频率为

$$f_{\max} = \frac{1}{t_{p(\text{总时间})}} = \frac{1}{40\ \text{ns}} = 25\ \text{MHz}$$

相关问题： 如果图 8.5(a)中的所有触发器都是上升沿触发，画出时序图。

8.1.3 异步译码计数器

计数器的模是计数器按顺序经过的独特状态的数目。计数器的最大可能状态数(最大模)

是 2^n，其中 n 是计数器内部的触发器个数。当然，可以设计计数器，使得其序列中的状态个数小于最大值 2^n。这种序列类型称为截断序列。

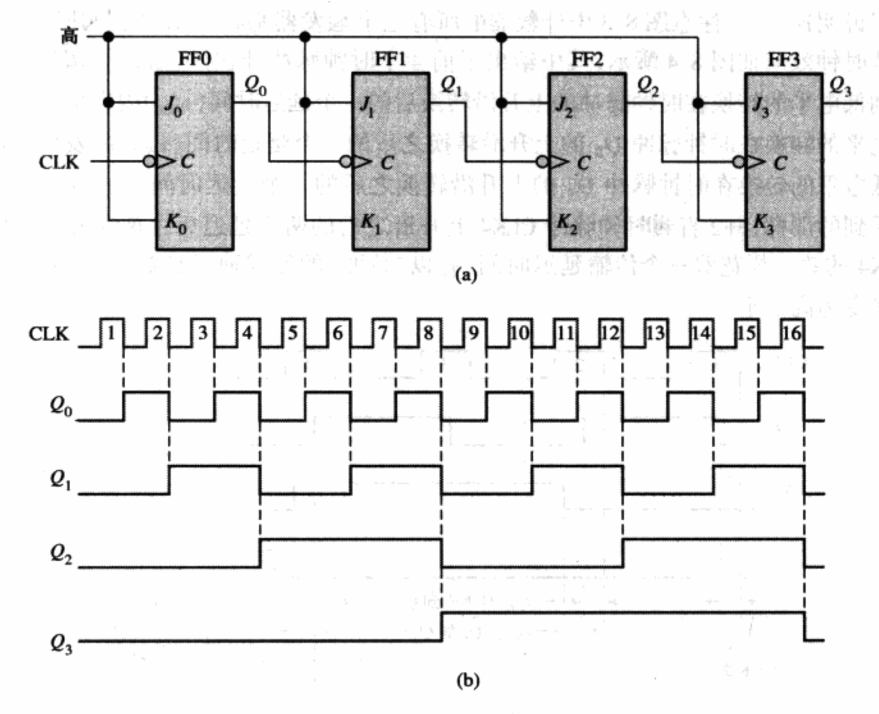


图 8.5　4 位异步二进制计数器和时序图

◇ *计数器可以有 2^n 个状态，其中 n 是触发器的个数。*

一个具有截断序列的计数器的常见模是 10(称为模 10)。序列中具有 10 个状态的计数器称为十进制计数器。具有从 0(0000)到 9(1001)计数序列的十进制计数器是 BCD 十进制计数器，因为它的 10 种状态序列可以产生 BCD 码。这种类型的计数器在一些显示应用中很有用，使用中需要将 BCD 码变换为十进制读数。

为了得到截断的序列，需要迫使计数器在经过所有可能的状态之前再次循环。例如，BCD 十进制计数器必须在 1001 状态后再循环回到 0000 状态。一个十进制计数器需要四个触发器(三个触发器是不够用的，因为 $2^3 = 8$)。

使用一个如例 8.1 中那样的 4 位异步计数器，并修改它的序列以解释截断计数器的原理。一种使得计数器在计数到 9(1001)之后再循环的方法是，用一个与非门对计数值 10(1010)译码，并把此与非门的输出连接到触发器的清零 $\overline{CLR}$ 输入上，如图 8.6(a)所示。

部分译码　注意在图 8.6(a)中，只有 Q_1 和 Q_3 连接到了与非门的输入上。这种安排是部分译码的一个例子，其中两个独特的状态($Q_1 = 1$ 和 $Q_3 = 1$)就足以对计数值 10 进行译码，因为其他所有状态(0～9)都不会同时具有高电平 Q_1 和高电平 Q_3。当该计数器进入计数值 10(1010)时，译码门输出就会变为低电平并且使所有的触发器异步复位。

结果时序图如图 8.6(b)所示。注意在 Q_1 波形上有一个假信号。产生假信号的原因是 Q_1 必须在计数值 10 被译码之前首先变为高电平。直到该计数器进入计数值 10 以后的几纳

秒，译码门的输出才会变为低电平（两个输入都是高电平）。因此，在复位到 0000 之前，计数器在 1010 状态上停留一个较短的时间，因而产生了 Q_1 上的假信号，以及用来复位计数器的 $\overline{CLR}$ 线上传来的假信号。

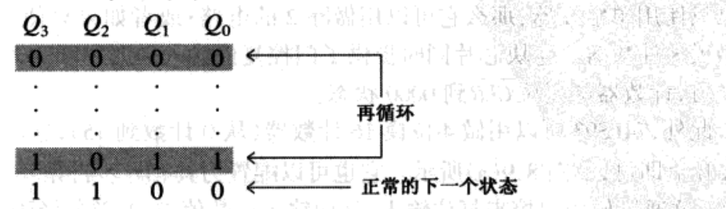
其他的截断序列可以用相似的方式实现，如例 8.2 所示。

图 8.6　具有异步再循环的异步触发的十进制计数器

例 8.2　给出怎样实现一个异步计数器，使其模为 12，具有从 0000 到 1011 的直接二进制序列。

解：由于三个触发器可以产生最多 8 个状态，所以就需要四个触发器来产生任何大于 8 而小于或者等于 16 的模。

当该计数器进入它的最后一个状态 1011 后，它就必须再循环到 0000 而不是进入其正常的下一个状态 1100，如下面的序列图所示：



观察 Q_0 和 Q_1 都变为 0,但是 Q_2 和 Q_3 都必须在第 12 个时钟脉冲时被迫转换为 0。如图 8.7(a)给出了这个模 12 计数器。与非门部分译码计数值 12(1100)并且使触发器 2 和触发器 3 复位。因此,在第 12 个时钟脉冲时,计数器被迫从计数值 11 再循环回到计数值 0,如图 8.7(b)中的时序图所示。(在被 $\overline{CLR}$ 上的假信号复位之前,计数器只在计数值 12 上停留几纳秒的时间。)

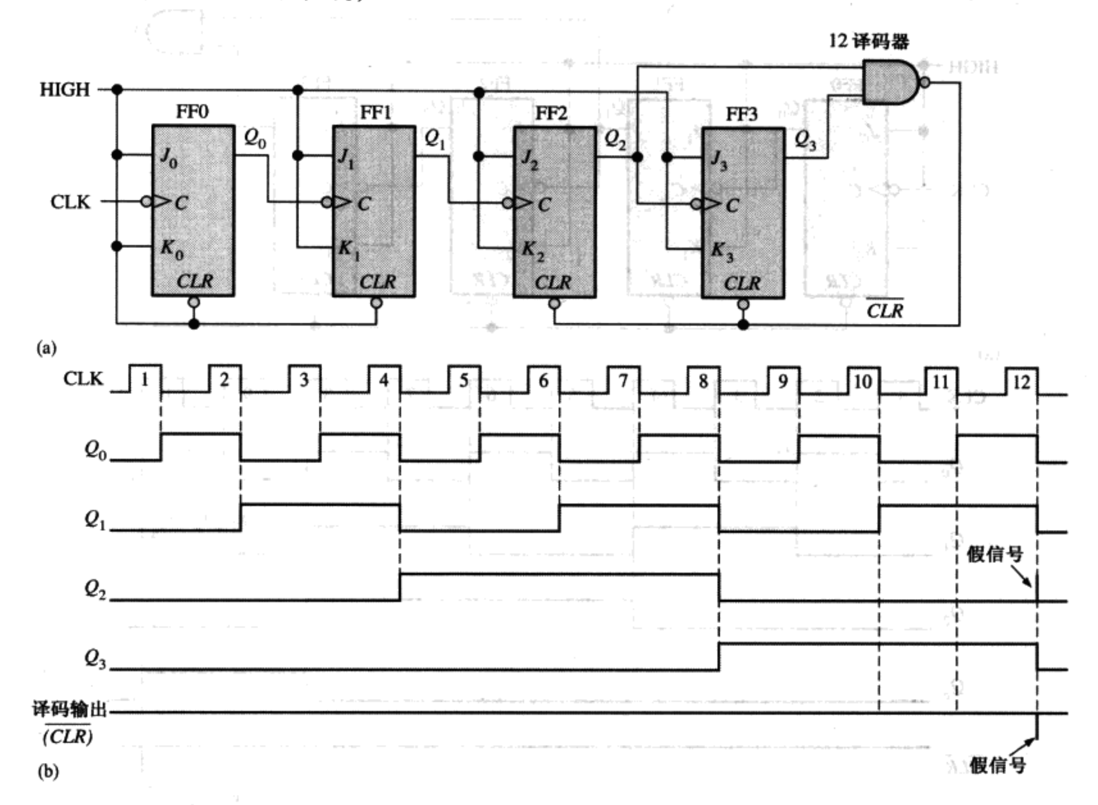


图 8.7　具有异步再循环的异步触发的模 12 计数器

相关问题:怎样修改图 8.7(a)中的计数器使其变为模 13 计数器?

74LS93 4 位异步二进制计数器

74LS93 是特定集成电路异步计数器的一个例子。如图 8.8 中逻辑图所给出的那样,这种芯片实际上由一个单触发器和一个 3 位异步计数器组成的,这种安排是出于灵活操作的目的。如果只使用单触发器,那么它可以用做除 2 的电路;或者如果只使用 3 位计数器,那么它可以用做模 8 计数器。这块芯片同时提供了门控复位输入 *RO*(1)和 *RO*(2)。当这两个输入都是高电平时,计数器就复位 $\overline{CLR}$ 到 0000 状态。

此外,74LS93 可以用做 4 位模 16 计数器(从 0 计数到 15),这通过把输出 Q_0 连接到输入 CLK B 上即可,如图 8.9(a)所示。它也可以配置为具有异步再循环的十进制计数器(从 0 计数到 9),这通过使用门控来复位输入,从而输入计数值 10 的部分译码,如图 8.9(b)所示。

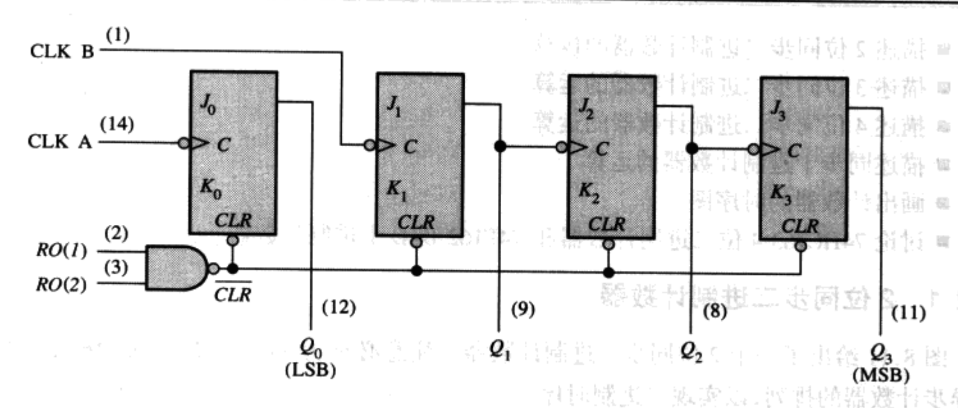


图 8.8 74LS93 4 位异步二进制计数器逻辑图(引脚编号在圆括号内,并且所有的J和K输入都内部连接为高电平)

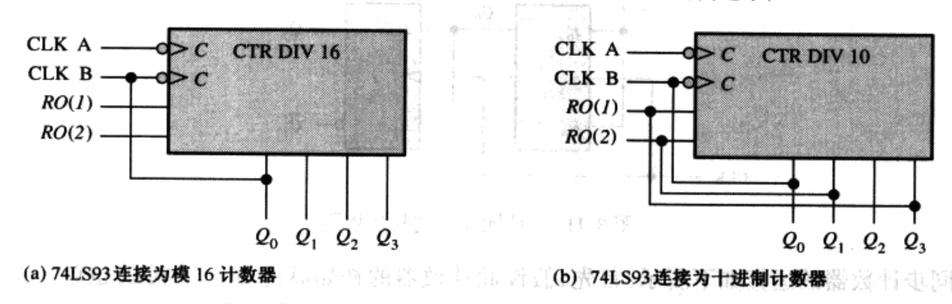


图 8.9 74LS93 异步计数器的两种配置(限制符号 CTR DIV n 表示具有 n 个状态的计数器)

例 8.3 描述 74LS93 怎样用做模 12 计数器。

解:使用门控复位输入 $RO(1)$和 $RO(2)$,实现部分译码计数值 12(记住,有一个内部与非门和这两个输入关联)。计数值 12 的译码通过把 Q_3 连接到 $RO(1)$并把 Q_2 连接到 $RO(2)$来完成,如图 8.10所示。输出 Q_0 连接到 CLK B 上,最后形成一个 4 位计数器。

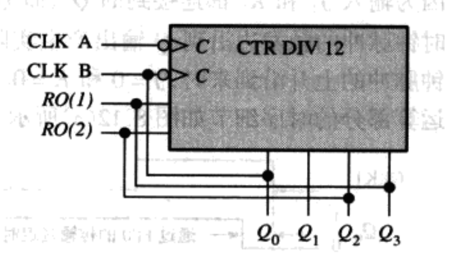


图 8.10 74LS93 连接为模 12 计数器

在计数器进入计数值 12(1100)之后的瞬间,它就会复位到 0000。但是,这个再循环会在 Q_2 上产生一个假信号,因为该计数器在再循环之前,必须进入 1100 状态并停留几纳秒的时间。

相关问题:给出 74LS93 怎样连接为模 13 计数器。

8.2 同步计数器运算

术语同步是指事件相互之间具有固定的时间关系。同步计数器就是计数器中所有的触发器由一个共同的时钟脉冲在相同的时间触发。

学完本节以后,应当能够

- 描述 2 位同步二进制计数器的运算
- 描述 3 位同步二进制计数器的运算
- 描述 4 位同步二进制计数器的运算
- 描述同步十进制计数器的运算
- 画出计数器的时序图
- 讨论 74HC163 4 位二进制计数器和 74F162 BCD 十进制计数器

8.2.1 2 位同步二进制计数器

图 8.11 给出了一个 2 位同步二进制计数器。注意必须为 FF1 的 J_1 和 K_1 输入使用不同于异步计数器的排列,以实现二进制时序。

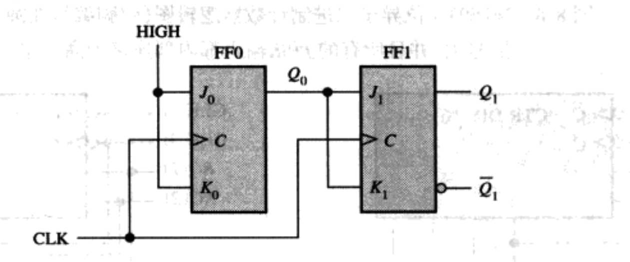


图 8.11 2 位同步二进制计数器

同步计数器的运算如下所示:首先,假设此计数器的初始状态是二进制 0,也就是两个触发器都是复位状态。当第一个时钟脉冲的上升沿来到时,FF0 将切换,因而 Q_0 变为高电平。那么 FF1 在 CLK1 的上升沿到来时会发生什么情况呢?为了找出答案,考虑 FF1 的输入条件。因为输入 J_1 和 K_1 都连接到的 Q_0,而 Q_0 还没有低电平,所以 J_1 和 K_1 都是低电平。记住,从时钟脉冲的触发边沿到 Q 输出产生实际上的转换,这之间存在传输延迟。所以,当第一个时钟脉冲的上升沿到来时,$J=0$ 和 $K=0$。这是个无变化情况,所以 FF1 没有改变状态。计数器运算部分的时序细节如图 8.12(a)所示。

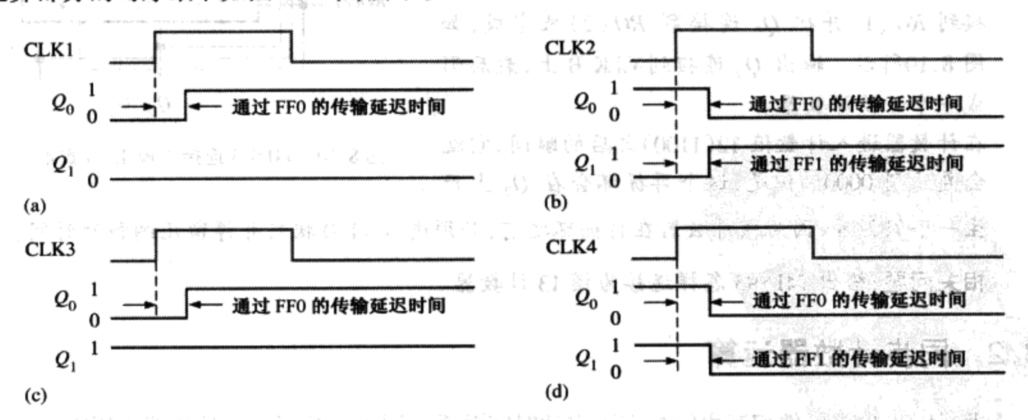


图 8.12 2 位同步计数器运算部分的时序细节(两个触发器的传输延迟时间假设相等)

◇ 在同步计数器中时钟脉冲作用在每一个触发器上。

在 CLK1 之后，$Q_0 = 1$ 和 $Q_1 = 0$（这是二进制 1 状态）。当 CLK2 的上升沿到来时，FF0 将切换，并且 Q_0 将变为低电平。由于 FF1 在这个时钟脉冲的触发边沿上时，输入 J_1 和 K_1 都是高电平（$Q_0 = 1$），所以触发器切换，并且 Q_1 变为高电平。因此，在 CLK2 之后，$Q_0 = 0$ 和 $Q_1 = 1$（这是二进制 2 状态）。这个情况的时序细节如图 8.12(b)所示。

当 CLK3 的上升沿来到时，FF0 再次切换到置位状态（$Q_0 = 1$），而 FF1 保持置位（$Q_1 = 1$），因为它的 J_1 和 K_1 输入都是低电平（$Q_0 = 0$）。在这个触发边沿之后，$Q_0 = 1$ 和 $Q_1 = 1$（这是二进制 3 状态）。时序细节如图 8.12(c)所示。

最后，在 CLK4 的上升沿，Q_0 和 Q_1 变为低电平，因为在它们的 J 和 K 输入上都有切换情况。时序细节如图 8.12(d)所示。这时计数器再循环回到它的原始状态：二进制 0。

图 8.11 中计数器完整的时序图如图 8.13 所示。注意所有的波形转换看起来都是同时发生的，也就是传输延迟没有指示出来。虽然延迟在同步计数器运算中是一个重要的因素，但是在整个时序图中出于简化的目的把它们省略了。即使没有给出很短的延迟和非常小的时序差别，电路正常运算所产生的主要的波形关系也可以完全表达出来。然而，在高速数字电路中，这些小的延迟在设计和故障检测中是一个重要的考虑因素。

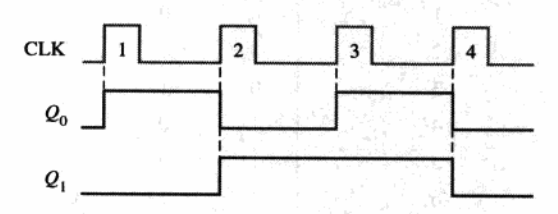


图 8.13 图 8.11 的计数器时序图

8.2.2 3位同步二进制计数器

一个 3 位同步二进制计数器如图 8.14 所示，它的时序图如图 8.15 所示。通过观察表 8.3 所示的状态时序，就可以理解这个计数器的运算。

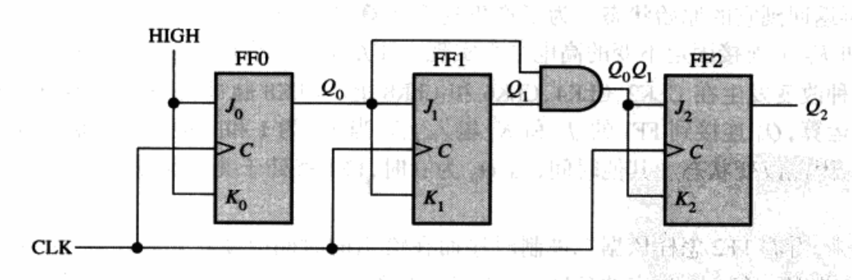


图 8.14 3位同步二进制计数器



计算机小知识

奔腾处理器中的 TSC（时间印章计数器）用来进行性能监测，它可以使得一些对于奔腾系统的整体性能很重要的参数能够正确地确定下来。通过阅读一个程序执行前后的 TSC，基于处理器循环时间就可以确定此程序所需要的精确时间。在这种方式中，TSC 构成了所有时间计算的基础，这些时间计算和系统运算的优化相

关。例如,可以准确确定两个或者多个程序序列中哪一个更加有效。这对于编译程序开发人员和系统程序员为奔腾处理器开发最有效的代码非常有用。

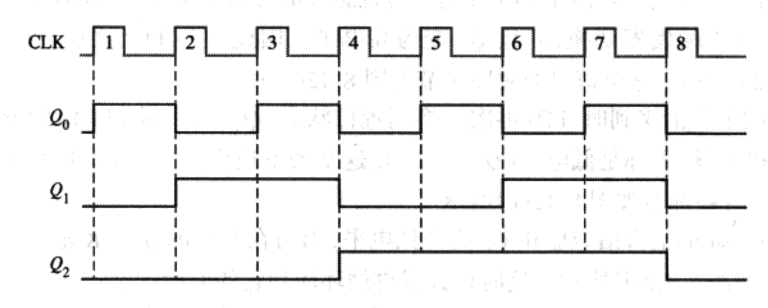


图 8.15 图 8.14 中计数器的时序图

表 8.3 3位二进制计数器的二进制状态时序

时钟输入	Q_2	Q_1	Q_0
初始状态	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8(再循环)	0	0	0

首先,观察 Q_0。注意到随着计数器从它的初始状态变化到最后一个状态然后再返回到初始状态,Q_0 在每个时钟到来时都改变状态。与此同时计数器从它的原始状态进入到最终状态,然后再返回到它的原始状态。为了产生这个运算,FF0 必须保持在切换模式下,这通过使输入 J_0 和 K_0 上连接固定不变的高电平来实现。注意 Q_1 在每次 Q_0 为 1 之后,都进入相反的状态。这种改变发生在 CLK2、CLK4、CLK6 和 CLK8 上。CLK8 脉冲使得计数器再循环。为了产生这个运算,Q_0 连接到 FF1 的 J_1 和 K_1 输入上。当 Q_0 为 1 和时钟脉冲到来时,FF1 就处于切换模式,因此改变状态。其他时间,当 Q_0 为 0 时,FF1 就处于无变化模式,并且保持它当前的状态。

接下来,看看 FF2 怎样依据二进制时序而在恰当的时间改变状态。注意 Q_2 的两次态改变,在这之前都会有一个特定的条件,即 Q_0 和 Q_1 都是高电平。这个条件由与门来检测,并且应用于 FF2 的 J_2 和 K_2 输入上。只要 Q_0 和 Q_1 都为高电平,与门的输出就会使得 FF2 的 J_2 和 K_2 输入为高电平,并且 FF2 在下一个时钟脉冲到来时切换。在所有的其他时间,J_2 和 K_2 输入都被与门输出保持为低电平,这样 FF2 的状态不改变。

8.2.3 4位同步二进制计数器

图 8.16(a)给出了一个 4 位同步二进制计数器,图 8.16(b)给出了它的时序图。这个特殊

的计数器由下降沿触发的触发器来实现。对于前三个触发器，J 和 K 输入控制的内在推导和前面所讨论的 3 位计数器是一样的。第四个阶段，FF3 在序列中只改变了两次。注意这两个转换的发生都在 Q_0、Q_1 和 Q_2 全部为高电平之后。这个情况由与门 G_2 来译码，使得在时钟脉冲到来时，FF3 就会改变状态。对于所有其他的时间，FF3 的 J_3 和 K_3 输入为低电平，并且处于无变化情况。

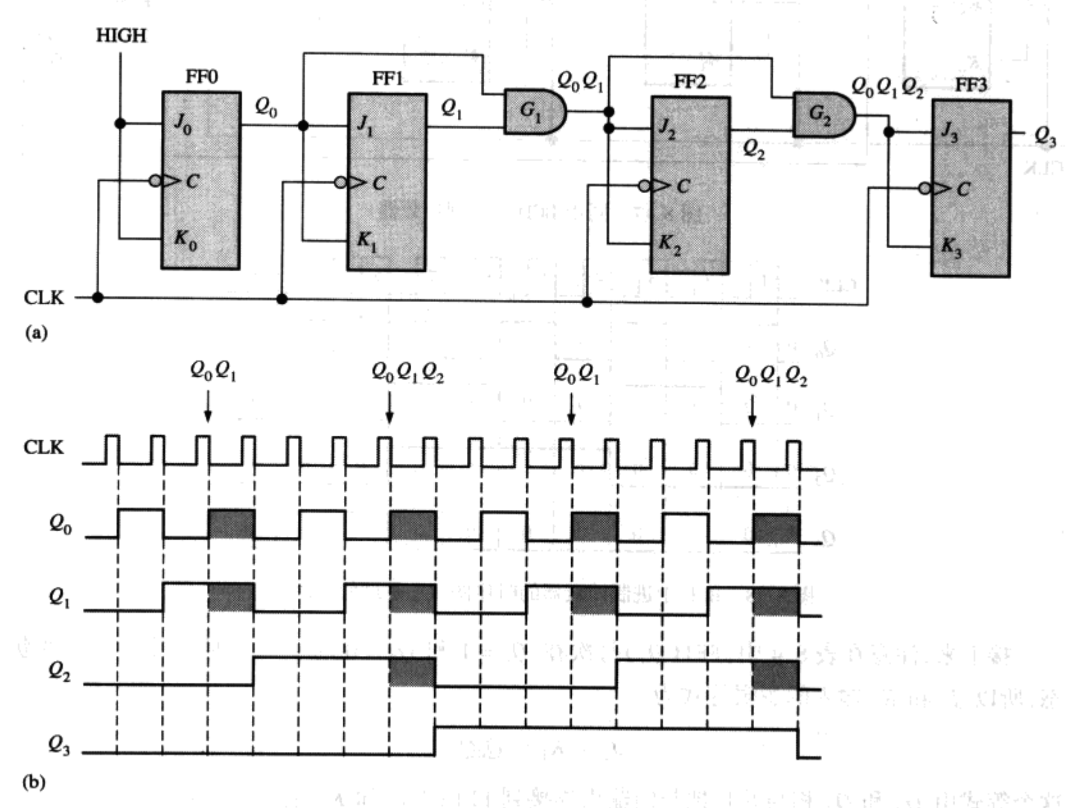


图 8.16　一个 4 位同步二进制计数器和时序图，与门输出为高电压的地方由阴影区域表示

8.2.4　4 位同步十进制计数器

正如我们所知道的那样，BCD 十进制计数器呈现出一个截断的二进制序列，从 0000 到 1001 状态，再从 1001 状态循环回到 0000 状态，而不是从 1001 状态到 1010 状态。同步 BCD 十进制计数器如图 8.17 所示。十进制计数器的时序图如图 8.18 所示。

◇ 十进制计数器具有 10 个状态。

通过检查表 8.4 中的状态序列并且遵循图 8.17 中的实现方法，就可以理解该计数器的运算。首先，注意 FF0(Q_0)在每一个时钟脉冲到来时切换，所以 J_0 和 K_0 输入的逻辑等式为

$$J_0 = K_0 = 1$$

这个等式通过把 J_0 和 K_0 连接到固定不变的高电平上来实现。

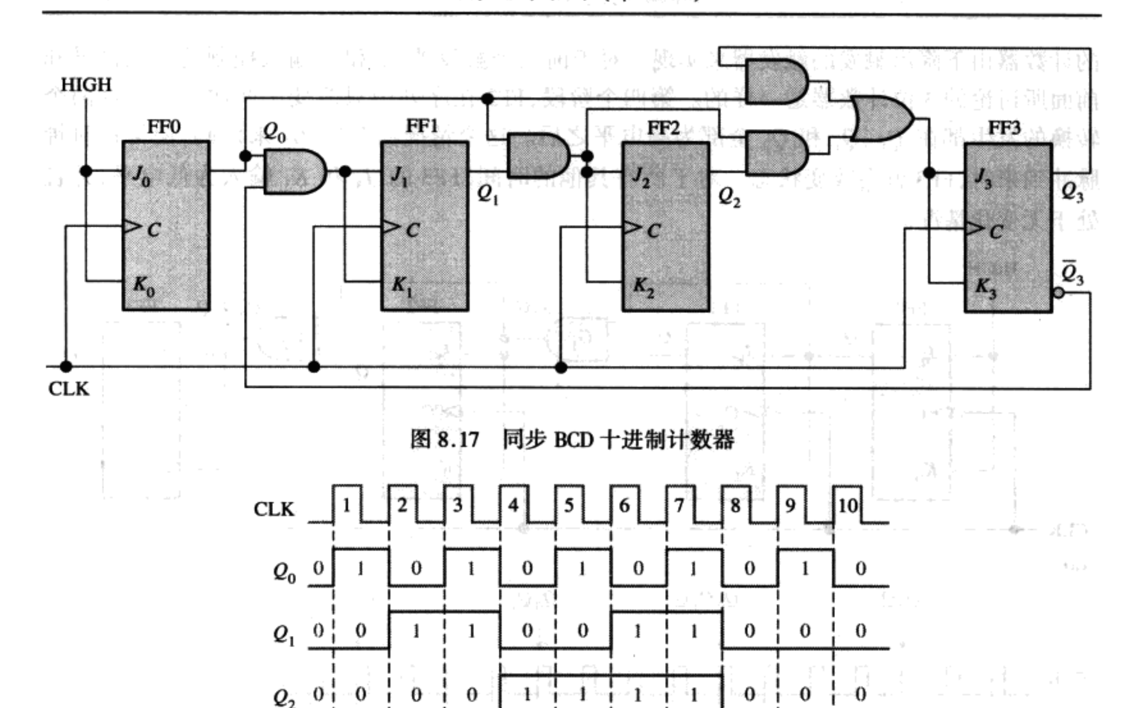


图 8.17　同步 BCD 十进制计数器

图 8.18　BCD 十进制计数器的时序图(Q_0 是最低有效位)

接下来,注意在表 8.4 中,FF1(Q_1)每次在 $Q_0 = 1$ 和 $Q_3 = 0$ 的下一个时钟脉冲上改变状态,所以 J_1 和 K_1 输入的逻辑等式为

$$J_1 = K_1 = Q_0\overline{Q}_3$$

这个等式由 Q_0 和 $\overline{Q}_3$ 相与并且把与门输出连接到 FF1 的 J_1 和 K_1 输入上来实现。

触发器 2(Q_2)每次在 $Q_0 = 1$ 和 $Q_1 = 1$ 的下一个时钟脉冲改变状态。这需要如下所示的输入逻辑等式:

$$J_2 = K_2 = Q_0Q_1$$

这个等式由 Q_0 和 Q_1 相与并且把门输出连接到 FF2 的 J_2 和 K_2 输入上来实现。

最后,FF3(Q_3)每次在 $Q_0 = 1$、$Q_1 = 1$ 和 $Q_2 = 1$(状态 7)的下一个时钟脉冲变为相反的状态,或者在 $Q_0 = 1$ 和 $Q_3 = 1$(状态 9)时改变。所对应的等式如下所示:

$$J_3 = K_3 = Q_0Q_1Q_2 + Q_0Q_3$$

这个函数由连接到 FF3 的 J_3 和 K_3 输入上的与/或逻辑来实现,如图 8.17 中的逻辑图所示。注意这个十进制计数器和图 8.16 中的模 16 二进制计数器的区别是 $Q_0\overline{Q}_3$ 与门、Q_0Q_3 与门和或门;这种安排检测 1001 状态的发生,并且使得此计数器在下一个时钟脉冲到来时正确地再循环。

表 8.4 BCD 十进制计数器的状态

时钟输入	Q_3	Q_2	Q_1	Q_0
初始状态	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10（再循环）	0	0	0	0

74HC163 4位同步二进制计数器

74HC1 63 是集成电路4位同步二进制计数器的一个例子。逻辑符号如图 8.19 所示，引脚编号在圆括号内。除了前面所讨论的一般的同步二进制计数器具有的基本功能之外，这种计数器还具有几个特征。

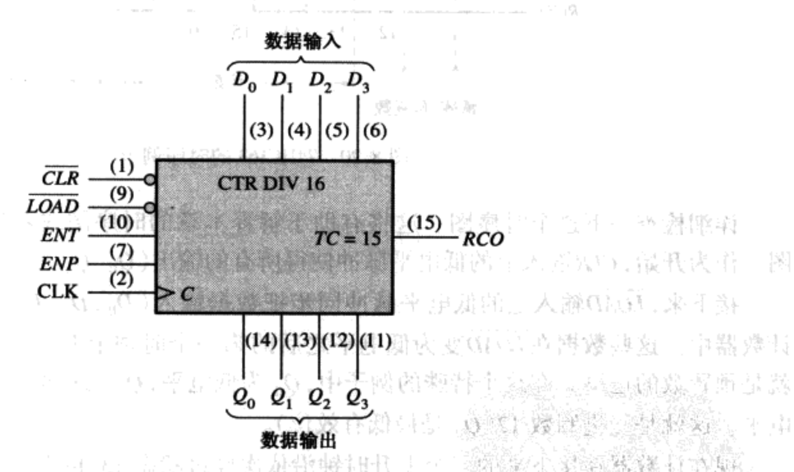


图 8.19 74HC163 4位同步二进制计数器(限制符号 CTR DIV 16 表示具有 16 个状态的计数器)

首先，通过把恰当的电平加到并行数据输入，计数器可以被同步预置到任意的4位二进制数。当低电平加到$\overline{LOAD}$(置数)输入时，计数器将在下一个时钟脉冲到来时读取数据输入的状态。因此，该计数器时序可以开始于任何4位二进制数。

同样，存在一个有效低电平清零输入$\overline{CLR}$，其同步复位计数器中所有的四个触发器。有两个使能输入 *ENP* 和 *ENT*。这两个输入必须都为高电平，才能使计数器顺序经过它的二进制状态。只要有一个输入为低电平时，计数器就不工作。当计数器到达序列 15 的最后一种状态

时,异步(行波)时钟输出(*RCO*)就变为高电平,这称为终端计数(*TC* = 15)。这个输出(*RCO*)和使能输入(*ENP* 和 *ENT*)一起,使得计数器可以串接起来,从而得到更大的计数序列。

图 8.20 给出了这个计数器预置为 12(1100)和计数到最后一位 15(1111)的时序图。输 D_0 是最低有效输入位,Q_0 是最低有效输出位。

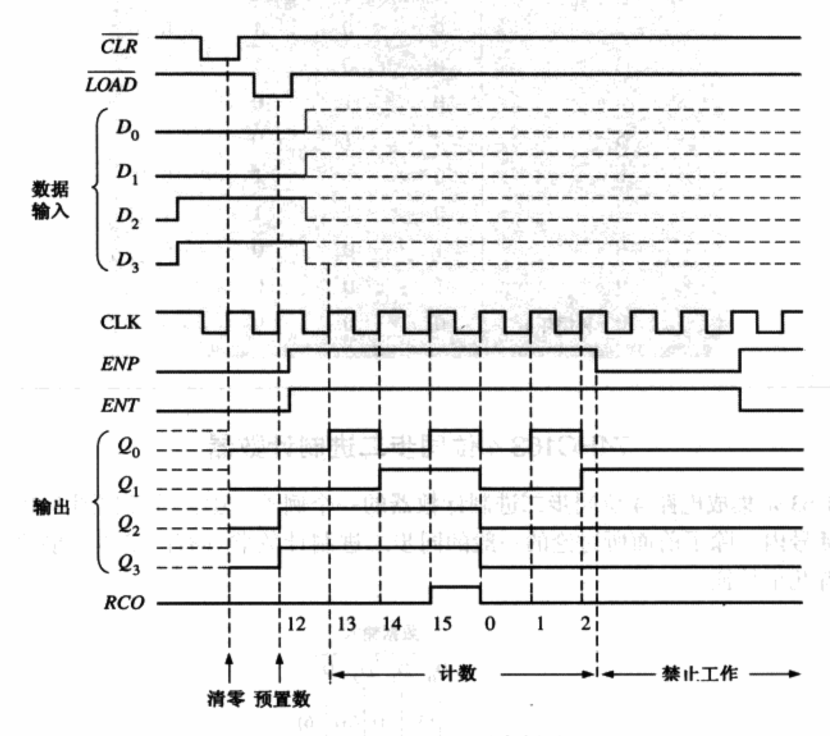


图 8.20　74HC163 的时序例子

详细检查一下这个时序图。这将有助于解释本章的时序图或者生产商数据表上的时序图。作为开始,$\overline{CLR}$输入上的低电平脉冲使得所有的输出(Q_0、Q_1、Q_2 和 Q_3)变为低电平。

接下来,$\overline{LOAD}$输入上的低电平脉冲同步把数据输入(D_0、D_1、D_2 和 D_3)上的数据输入到计数器中。这些数据在$\overline{LOAD}$变为低电平之后的第一个时钟上升沿时,出现在 *Q* 输出上。这就是预置数的运算。在这个特殊的例子中,Q_0 为低电平,Q_1 为低电平,Q_2 为高电平,Q_3 为高电平。这就是二进制数 12(Q_0 是最低有效位)。

现在计数器在接下来的三个上升时钟沿依次经过状态 13、14 和 15。然后在接下来的时钟脉冲下再循环回到 0、1、2。注意,*ENP* 和 *ENT* 输入在状态时序期间都是高电平。当 *ENP* 为低电平时,计数器就会被禁止工作并且保持在二进制数 2 的状态。

74F162 同步 BCD 十进制计数器

74F162 是十进制计数器的一个例子,使用数据输入端及$\overline{PE}$(预置数)输入上是低电平,计数器可以被预置到任意的 BCD 计数值。同步$\overline{SR}$(同步复位)上的低电平使计数器复位(清零)。使能输入端 *CEP* 和 *CET* 必须都是高电平,因为计数器响应时钟 CLK 输入的上升沿转换时,能

够使计数器经过它的顺序状态。使能输入和终端输出 *TC*(1001)一起,提供了几个十进制计数器级联的方法。图 8.21 给出了 74F162 计数器的逻辑符号,图 8.22 是计数器从复位到计数值 7(0111)的时序图。级联计数器将在 8.5 节讨论。

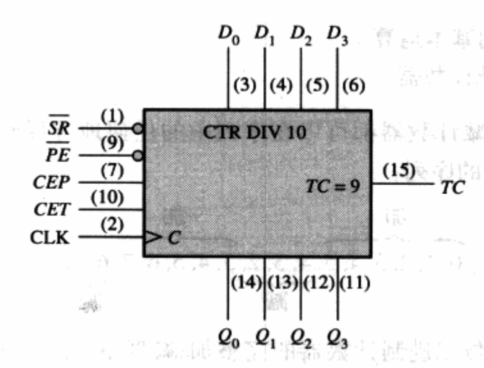


图 8.21 74F162 同步 BCD 十进制计数器(限制符号 CTR DIV 10 表示具有 10 个状态的计数器)

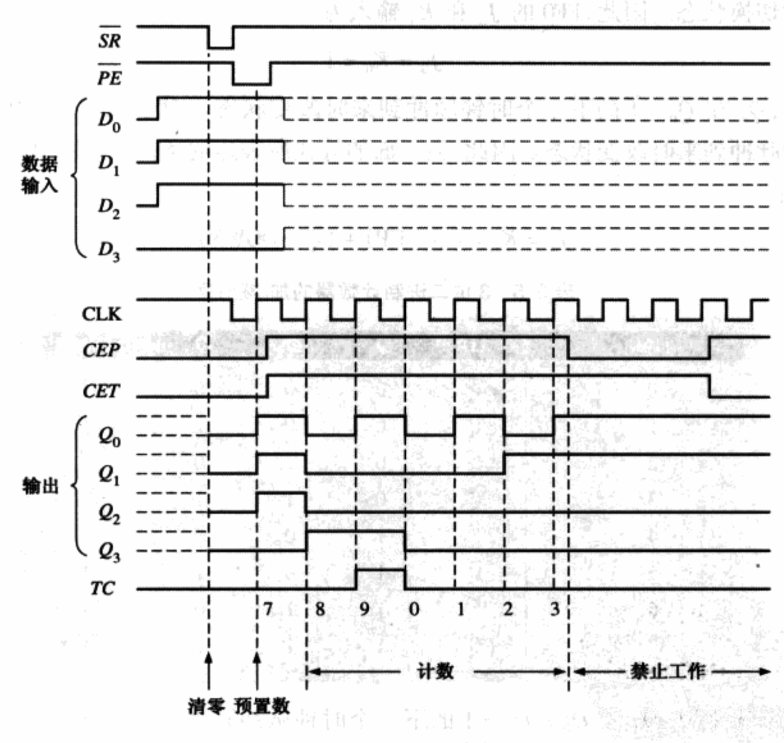


图 8.22 74F162 的时序例子

8.3 加/减同步计数器

加/减计数器能够以两个方向经过某一时序。加/减计数器有时候称为双向计数器,可以

具有任意指定的状态时序。一个3位二进制计数器向前经过它的时序(0,1,2,3,4,5,6,7),然后可以反过来,使得它以从相反的方向经过此时序(7,6,5,4,3,2,1,0)。

学完本节以后,应当能够

- 解释加/减计数器的基本运算
- 讨论74HC190加/减计数器

一般来说,大多数加/减计数器都可以在序列中的任何地方反转。例如,3位二进制计数器可以做到经过如下所示的序列:

$$\overbrace{0,1,2,3,4,5,}^{\text{加}}\underbrace{4,3,2,}_{\text{减}}\overbrace{3,4,5,6,7,}^{\text{加}}\underbrace{6,5,}_{\text{减}}\text{等}$$

表8.5给出了一个3位二进制计数器的完整加/减时序。箭头表示计数器在加和减的两种运算模式下所进行的状态到状态的运动方向。检查加和减序列的 Q_0,指出FF0在每个时钟脉冲到来时切换状态。因此,FF0的 J_0 和 K_0 输入为

$$J_0 = K_0 = 1$$

对于加时序,Q_1 在 $Q_0 = 1$ 的下一个时钟脉冲到来时改变状态。对于减时序,Q_1 在 $Q_0 = 0$ 的下一个时钟脉冲到来时改变状态。因此,在下面的等式所表达的条件下,FF1的 J_1 和 K_1 输入必须等于1:

$$J_1 = K_1 = (Q_0 \cdot \text{UP}) + (\overline{Q}_0 \cdot \text{DOWN})$$

表8.5　3位二进制计数器的加/减时序

时钟输入	UP	Q_2	Q_1	Q_0	DOWN
0	↑	0	0	0	↓
1	↑	0	0	1	↓
2	↑	0	1	0	↓
3	↑	0	1	1	↓
4	↑	1	0	0	↓
5	↑	1	0	1	↓
6	↑	1	1	0	↓
7	↑	1	1	1	↓

对于加时序来说,Q_2 在 $Q_0 = Q_1 = 1$ 的下一个时钟脉冲到来时改变状态。对于减时序来说,Q_2 在 $Q_0 = Q_1 = 0$ 的下一个时钟脉冲到来时改变状态。因此,在下面的等式所表达的条件下,FF2的 J_2 和 K_2 输入必须等于1:

$$J_2 = K_2 = (Q_0 \cdot Q_1 \cdot \text{UP}) + (\overline{Q}_0 \cdot \overline{Q}_1 \cdot \text{DOWN})$$

每个触发器的 J 和 K 输入对应的每一个条件,都会在计数器序列中的恰当位置产生一个切换。

如图 8.23 所示，对每个触发器的 J 和 K 输入使用上述逻辑等式，就是 3 位加/减二进制计数器的基本实现方法。注意 $UP/\overline{DOWN}$ 控制输入对于加是高电平，对于减是低电平。

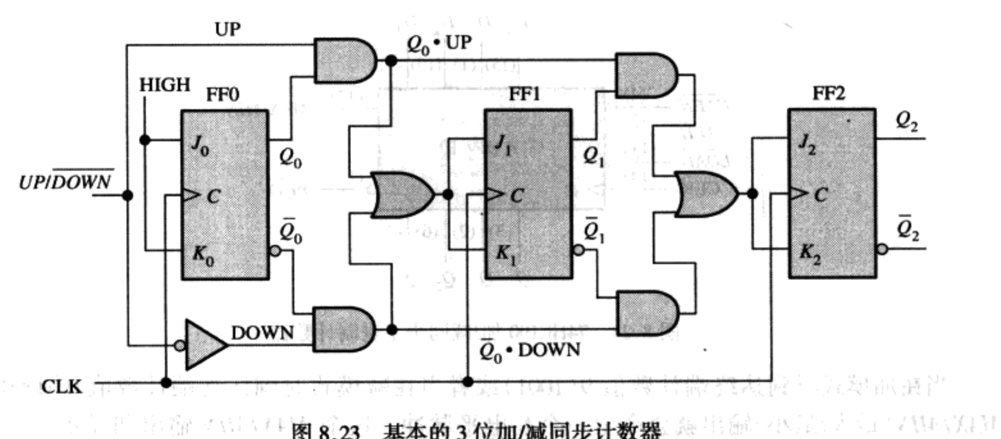


图 8.23　基本的 3 位加/减同步计数器

例 8.4　如果时钟和 $UP/\overline{DOWN}$（加/$\overline{减}$）控制输入具有如图 8.24(a) 所示的波形，给出时序图并确定 4 位同步二进制加/减计数器的时序。计数器开始于全 0 状态并且是上升沿触发。

解： 给出 Q 输出的时序图如图 8.24(b) 所示。根据这些波形，计数器序列如表 8.6 所示。

相关问题： 如果图 8.24(a) 的 $UP/\overline{DOWN}$ 控制波形反相，请给出时序图。

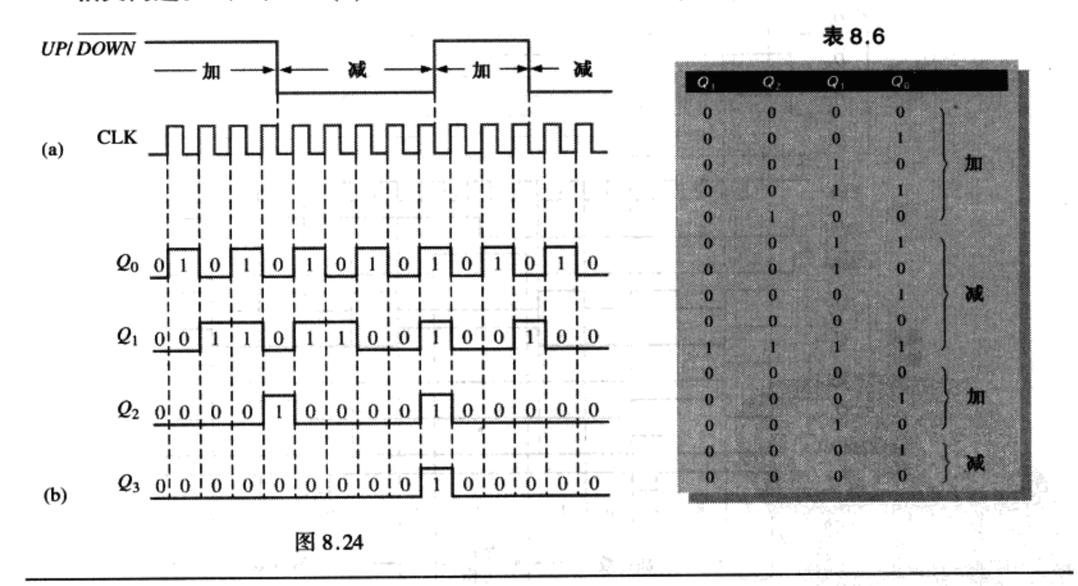


表 8.6

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	加
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	0	1	1	减
0	0	1	0	
0	0	0	1	
0	0	0	0	
1	1	1	1	
0	0	0	0	加
0	0	0	1	
0	0	1	0	
0	0	0	1	减
0	0	0	0	

图 8.24

74HC190 加/减十进制计数器

图 8.25 给出了 74HC190 的逻辑图，它是集成电路加/减同步计数器的一个例子。计数的方向由加/减输入（$D/\overline{U}$）的电平确定。当这个输入为高电平时，计数器为减计数；当它是低电

平时,计数器为加计数。同样,这个芯片可以被预置到任意所需的 BCD 数字,当$\overline{LOAD}$输入为低电平时,这个 BCD 数字由数据输入的状态来确定。

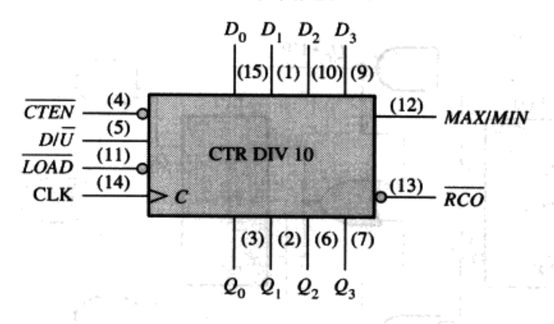


图 8.25　74HC190 加/减同步十进制计数器

当在加模式时到达终端计数值 9(1001)或者当在减模式时到达终端计数值 0(0000)后,*MAX/MIN*(最大/最小)输出就会产生一个高电平脉冲。这个 *MAX/MIN* 输出和异步(行波)时钟输出($\overline{RCO}$)及计数使能输入($\overline{CTEN}$)一起,用于级联计数器。(级联计数器将在 8.5 节讨论。)

图 8.26 是一个时序图,给出了 74HC190 计数器预置为 7(0111),然后经过一个加计数的时序并跟随着一个减计数的时序。当计数器位于全 0 状态(*MIN*)或者 1001 状态(*MAX*)时,*MAX/MIN* 输出就是高电平。

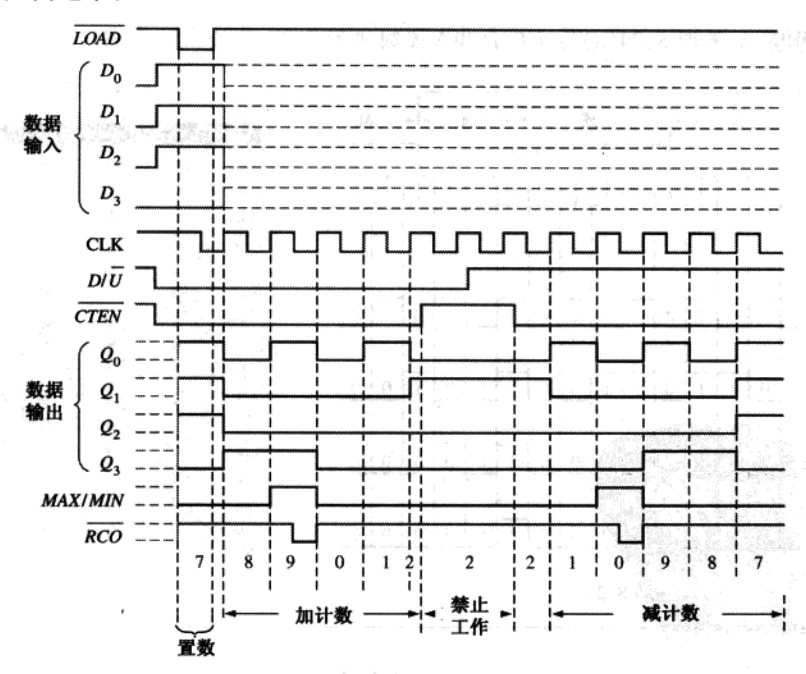


图 8.26　74HC190 的时序例子

8.4 同步计数器的设计

在这一节,将会看到时序电路设计技术在计数器设计中可以实现哪些特殊的应用。一般情况下,时序电路可以归为两类:

(1)其输出仅依赖于当前的内部状态(称为摩尔电路);

(2)其输出仅依赖于当前的内部状态和输入(称为米利电路)。

本节推荐给那些想要了解计数器设计或对状态机设计大体了解的读者。

学完本节以后,应当能够

- 根据基本部件和输入输出描述一般的序列电路
- 为给定的时序开发状态图
- 为指定的计数器时序开发次态表
- 创建触发器转换表
- 使用卡诺图方法推导同步计数器的逻辑需求
- 设计出一个计数器,用以产生指定的状态时序

8.4.1 时序电路的一般模式

在开始特定计数器设计技术之前,从时序电路或者状态机的一般性定义开始:一般的时序电路由组合逻辑电路和存储器(触发器)组成,如图 8.27 所示。在一个时钟触发的时序电路中,存储器部分有一个时钟输入,如图所示。

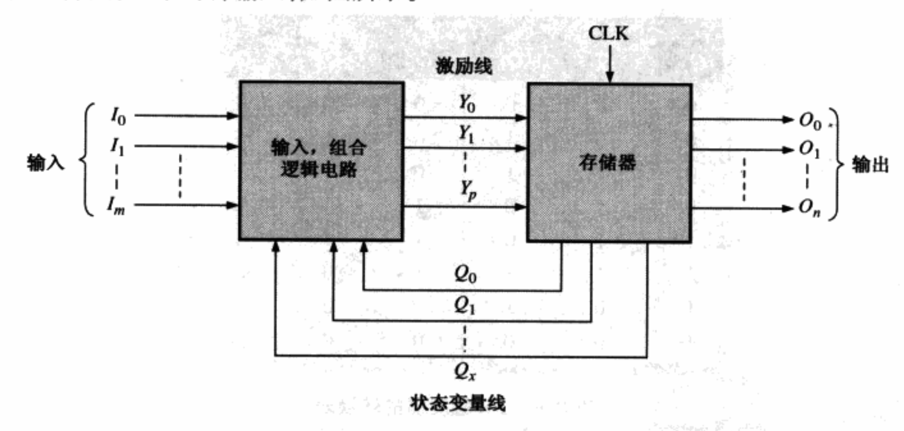


图 8.27 一般时钟触发的时序电路

电路的正常运算需要存储在存储器部分的信息和组合逻辑($I_0,I_1,\ldots,I_m$)的输入。在任何给定的时间,存储器所在的状态称为当前状态,并且将在时钟脉冲的作用下进入下一个状态,这个状态由激励线($Y_0,Y_1,\ldots,Y_P$)上的条件来确定。存储器的当前状态由状态变量($Q_0,Q_1,\ldots,Q_x$)来表示。这些状态变量和输入($I_0,I_1,\ldots,I_m$)一起确定了系统输出($O_0,O_1,\ldots,O_n$)。

并不是所有的时序电路都具有刚刚讨论的一般模式中所拥有的输入和输出组合。但是,所有的电路都具有激励变量和状态变量。计数器是时钟触发的时序电路的一个特例。在本节中,时序电路的一般设计过程将会在一系列步骤中应用于同步计数器。

8.4.2　步骤 1:状态图

设计计数器的第一步是创建一个状态图。一个状态图给出了计数器在时钟的作用下状态行进的历程。作为一个例子,图 8.28 给出了基本 3 位格雷码计数器的状态图。这个特殊电路除了时钟之外没有输入;除了计数器中每个触发器所取得的输出之外,没有其他输出。这时,可以复习一下第 2 章有关格雷码的内容。

8.4.3　步骤 2:次态表

一旦由状态图定义了时序电路,第二步就是推导出次态表,列出计数器的每一个状态(当前状态)和相应的下一个状态(次态)。次态就是计数器在时钟脉冲的作用下从当前状态转换过来的状态。次态表由状态图中导出,如表 8.7 所示的 3 位格雷码计数器。Q_0 是最低有效位。

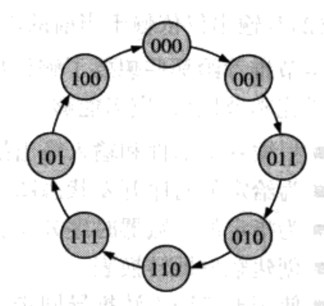


图 8.28　3 位格雷码计数器的状态图

8.4.4　步骤 3:触发器转换表

表 8.8 是 J-K 触发器的状态转换表。通过给出触发器从当前状态到次态的 Q 输出,列出了所有可能的输出转换。Q_N 是触发器的当前状态(在时钟脉冲之前),而 Q_{N+1}是次态(在时钟脉冲之后)。对于每个输出转换,列出了使得转换发生的输入 J 和K。X 表示“无关”(输入可以为 1 或者 0)。

表 8.7　3 位格雷码计数器的次态表

当前状态			次　态		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

表 8.8　J-K 触发器的转换表

输出转换			触发器输入	
Q_N		Q_{N+1}	J	K
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

Q_N: 当前状态
Q_{N+1}: 次态
X: “无关”

为了设计此计数器，基于次态表（见表8.7）的转换表应用于计数器中的每一个触发器。例如，对于当前状态000，Q_0 从当前状态0到次态1。为了使这种情况得以发生，J_0 必须为1，不要考虑 K_0 是什么（$J_0 = 1, K_0 = X$），正如可以在转换表（见表8.8）中看到的那样。接下来，Q_1 的当前状态为0并且在下一个状态保持为0，对于这个转换，$J_1 = 0, K_0 = X$。最后，Q_2 的当前状态为0而在下一个状态保持为0，所以，$J_2 = 0$ 和 $K_2 = X$。对于表8.7中的每一个当前状态，重复这样的分析。

8.4.5 步骤4：卡诺图

卡诺图可以用来确定计数器中每个触发器的 J 和 K 输入所需要的逻辑。对于每个触发器的 J 输入有一个卡诺图，每一个 K 输入也有一个卡诺图。在这个设计过程中，卡诺图中的每个小方格都表示列在表8.7计数器时序中的一个当前状态。

由转换表（见表8.8）中的 J 和 K 状态，1、0或者X置于卡诺图上每一个当前状态的小方格中，放什么取决于特定触发器 Q 输出的转换。为了解释这个过程，图8.29中给出了两个样例输入项，即最低有效位触发器 Q_0 的 J_0 和 K_0 输入。

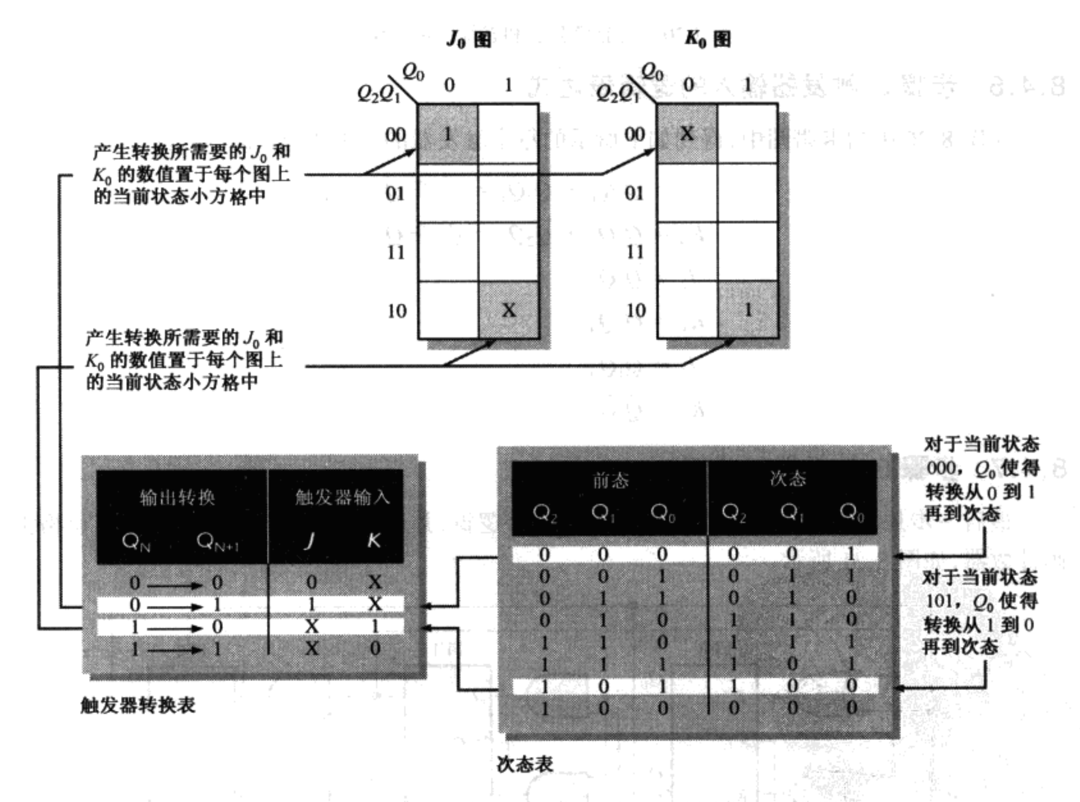


图8.29 表8.7和表8.8所表示的计数器时序映射过程的例子

计数器中所有三个触发器的完整卡诺图如图8.30所示。小方格的分组如图所示，并且推导了和每组相对应的布尔表达式。

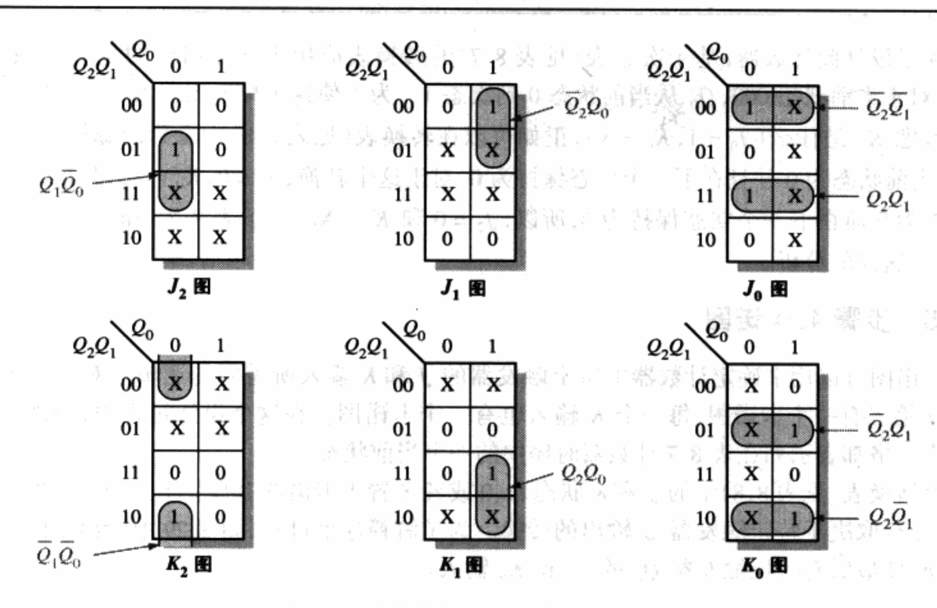


图 8.30 当前状态 J 和 K 输入的卡诺图

8.4.6 步骤 5:触发器输入的逻辑表达式

从图 8.30 中的卡诺图中,得到如下所示的每个触发器的 J 和 K 输入的表达式:

$$J_0 = Q_2Q_1 + \overline{Q}_2\overline{Q}_1 = \overline{Q_2 \oplus Q_1}$$
$$K_0 = Q_2\overline{Q}_1 + \overline{Q}_2Q_1 = Q_2 \oplus Q_1$$
$$J_1 = \overline{Q}_2Q_0$$
$$K_1 = Q_2Q_0$$
$$J_2 = Q_1\overline{Q}_0$$
$$K_2 = \overline{Q}_1\overline{Q}_0$$

8.4.7 步骤 6:计数器的实现

最后一步是从 J 和 K 输入表达式中实现组合逻辑,并连接触发器来构成完整的 3 位格雷码计数器,如图 8.31 所示。

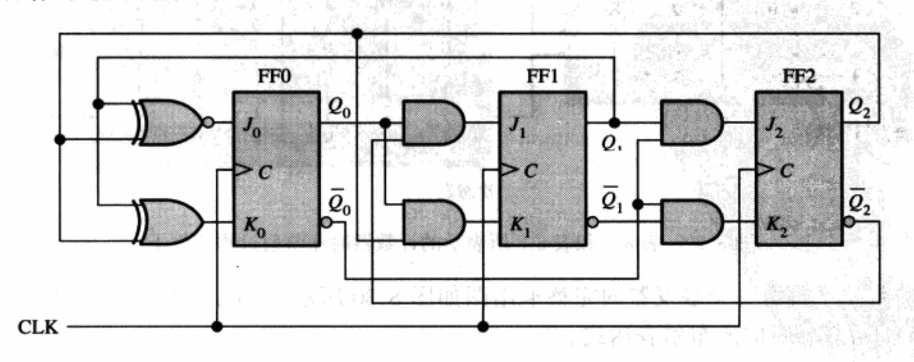


图 8.31 3 位格雷码计数器

设计计数器所需步骤的总结如下所示。一般来说,这些步骤可以应用于任何时序电路。

1. 确定计数器时序并绘制状态图。
2. 从状态图推导出次态表。
3. 开发转换表,给出每个转换所需要的触发器输入。转换表对于给定触发器的类型总是相同的。
4. 把 J 和 K 状态从转换表转移到卡诺图上。每个触发器的每个输入都有一个卡诺图。
5. 把卡诺图小方格分组,以产生并推导出每个触发器输入的逻辑表达式。
6. 使用组合逻辑实现表达式,并且连接触发器以创建计数器。

这个过程现在已应用于设计其他的同步计数器,如例 8.5 和例 8.6 所示。

例 8.5 为图 8.32 的状态图中所示的不规则二进制计数时序设计一个计数器。使用 J-K 触发器。

解:

步骤 1: 状态图如图 8.32 所示。虽然只有四个状态,但是需要一个 3 位计数器来实现这个时序,因为最大的二进制计数值是 7。由于所需要的时序并不包括所有可能的二进制状态,在设计中,无效状态(0、3、4 和 6)可以作为“无关”项处理。然而,如果计数器错误地进入某个无效状态,必须确保它返回有效状态。

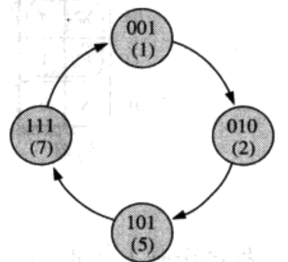


图 8.32

步骤 2: 从状态图中开发出来的次态表如表 8.9 所示。

表 8.9 次态表

当前状态			次 态		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	1	0	1	0
0	1	0	1	0	1
1	0	1	1	1	1
1	1	1	0	0	1

步骤 3: J-K 触发器的转换表在表 8.10 中给出。

表 8.10 J-K 触发器的转换表

输出转换			触发器输入	
Q_N		Q_{N+1}	J	K
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

步骤 4: J 和 K 输入绘制在图 8.33 中的当前状态卡诺图上。同样,“无关”项可以放置在相应的小方格中,这些小方格对应无效状态 000、011、100 和 110,由 X 表示。

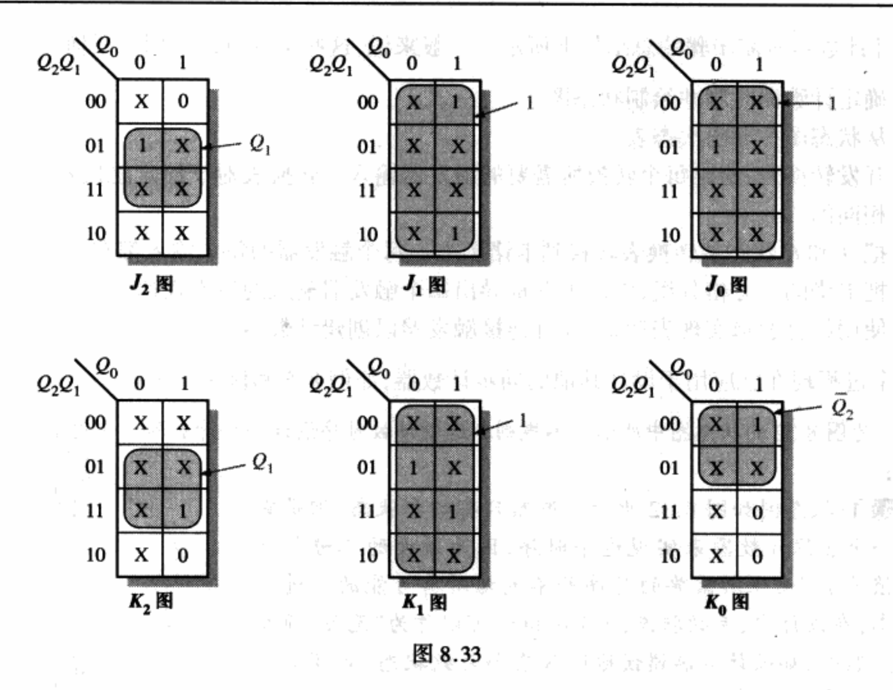


图 8.33

步骤 5: 将 1 分组,尽可能利用较多的"无关"项状态来实现最大的简化,如图 8.33 所示。注意当图中所有的小方格都被分组时,表达式就简单地等于 1。从图得到每个 J 和 K 输入的表达式,如下所示:

$$J_0 = 1, K_0 = \overline{Q}_2$$
$$J_1 = K_1 = 1$$
$$J_2 = K_2 = Q_1$$

步骤 6: 计数器的实现如图 8.34 所示。

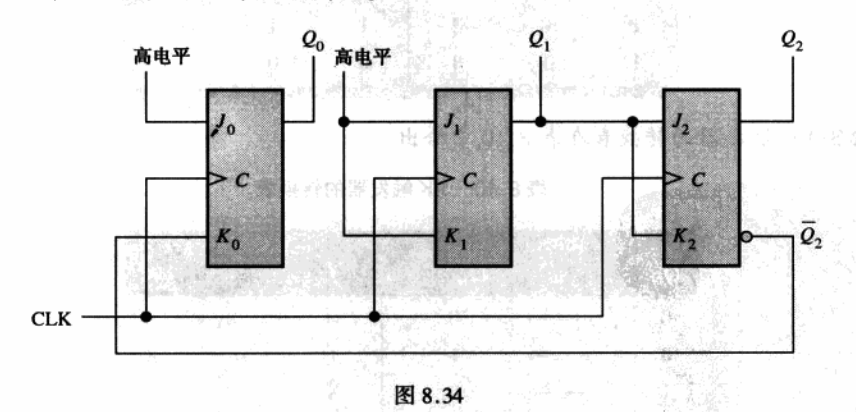


图 8.34

分析显示,如果该计数器偶尔进入无效状态(0,3,4,6)中的某一个状态,它总是能够返回到一个有效状态,根据下面的序列:0→3→4→7,6→1。

相关问题: 分析计数器总是能够从无效状态返回(最终)有效状态进行检验。

例 8.6 开发一个具有格雷码时序的3位加/减计数器。当 $UP/\overline{DOWN}$(加/减)控制输入为1时,该计数器应当加计数,而当控制输入为0时,应当减计数。

解:

步骤1: 状态图如图8.35所示。每个箭头旁边的1和0表示 $UP/\overline{DOWN}$ 控制输入 Y 的状态。

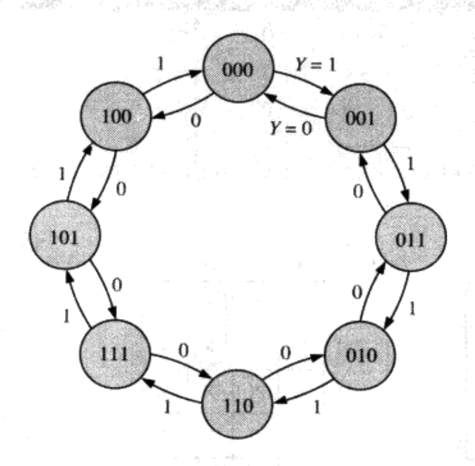


图8.35 3位加/减格雷码计数器的状态图

步骤2: 从状态图中导出的次态表如表8.11所示。注意对于每一个当前状态都有两个可能的次态,这取决于 $UP/\overline{DOWN}$ 控制变量 Y。

表8.11 位加/减格雷码计数器的次态表

当前状态			次态 Y = 0 (DOWN)			次态 Y = 1 (UP)		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

$Y = UP/\overline{DOWN}$ (加/减)控制输入

步骤3: J-K触发器的转换表重复于表8.12中。

步骤4: 触发器 J 和 K 输入的卡诺图如图8.36所示。控制输入 Y 及 Q_0、Q_1、Q_2 被考虑为

状态变量。使用次态表,将表 8.12 中"触发器输入"一列中的信息转移到卡诺图上,以表示计数器的每一个当前状态。

表 8.12　J-K 触发器的转换表

输出转换			触发器输入	
Q_N		Q_{N+1}	J	K
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

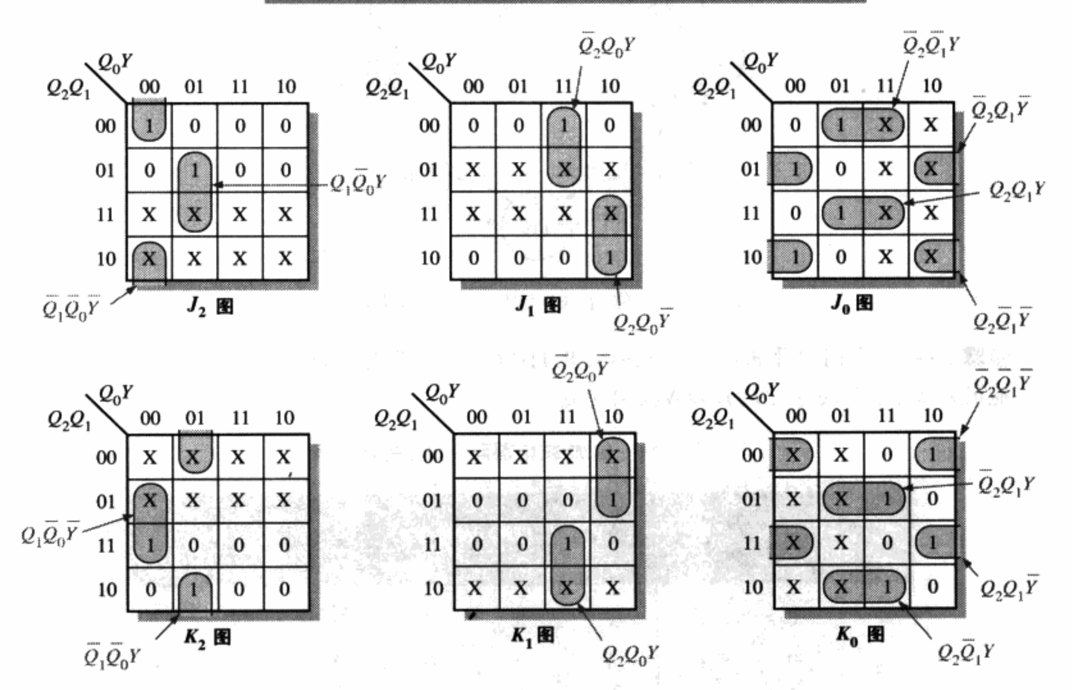


图 8.36　表 8.11 的 J 和 K 卡诺图,控制输入 Y 作为第四个变量

步骤 5:把 1 组合成尽可能大的组,只要可能就使用"无关"项。从这些组中分解出变量,J 和 K 输入的表达式如下所示:

$$J_0 = Q_2Q_1Y + Q_2\overline{Q}_1\overline{Y} + \overline{Q}_2\overline{Q}_1Y + \overline{Q}_2Q_1\overline{Y} \qquad K_0 = \overline{Q}_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1Y + Q_2\overline{Q}_1Y + Q_2Q_1\overline{Y}$$

$$J_1 = \overline{Q}_2Q_0Y + Q_2\overline{Q}_0\overline{Y} \qquad K_1 = \overline{Q}_2Q_0\overline{Y} + Q_2\overline{Q}_0Y$$

$$J_2 = Q_1\overline{Q}_0Y + \overline{Q}_1\overline{Q}_0\overline{Y} \qquad K_2 = Q_1\overline{Q}_0\overline{Y} + \overline{Q}_1\overline{Q}_0Y$$

步骤 6:使用组合逻辑实现 J 和 K 的等式,完整的计数器如图 8.37 所示。

相关问题:验证图 8.37 中的逻辑与步骤 5 中的表达式相符。

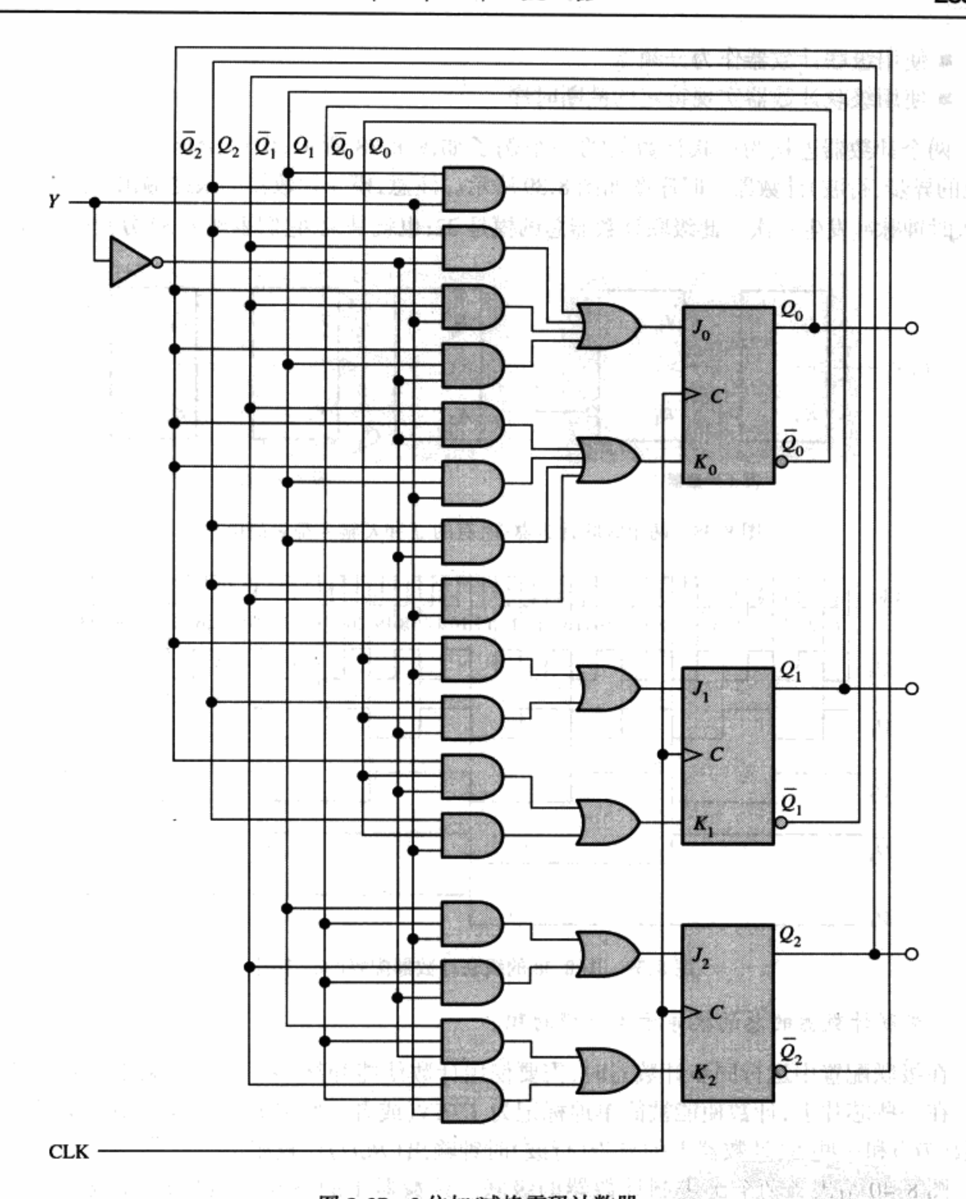


图 8.37　3 位加/减格雷码计数器

8.5　级联计数器

计数器可以按照级联的方式连接以实现更高模的运算。实质上，级联的意思就是一个计数器最后级的输出驱动下一个计数器的输入。

学完本节以后，应当能够

- 确定级联计数器的总的模
- 分析级联计数器配置的时序图

■ 使用级联计数器作为分频器
■ 使用级联计数器实现特定的截断时序

两个计数器连接为级联计数器的一个例子如图 8.38 所示,图中给出了一个 2 位和一个 3 位的异步(行波)计数器。时序图如图 8.39 所示。注意,模 8 计数器的最终输出 Q_4,每隔 32 个输入时钟脉冲发生一次。此级联计数器总的模是 32;也就是说,它们表现为 32 分频的计数器。

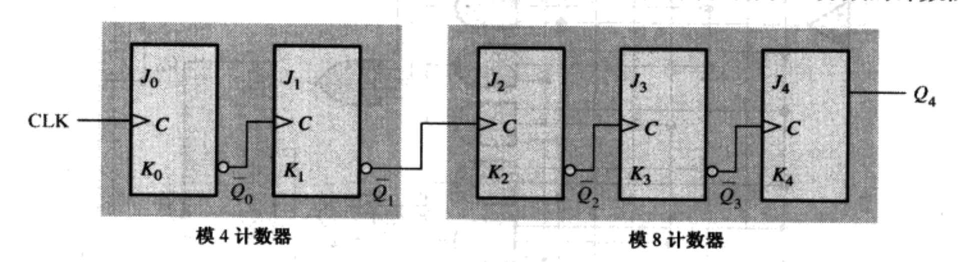


图 8.38 两个级联计数器(所有的 J 和 K 输入都是高电平)

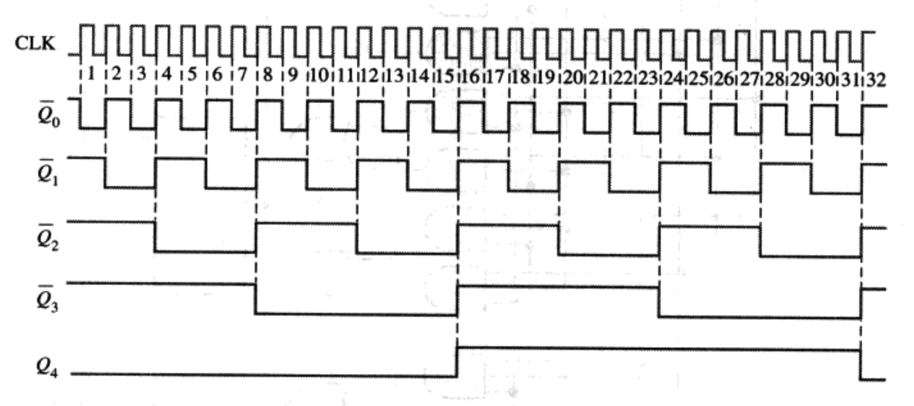


图 8.39 图 8.38 的级联计数器配置的时序图

◇ 级联计数器的总的模等于单个模的积。

在级联配置中运行同步计数器时,需要使用计数使能和终端计数功能来实现更高模的运算。在一些芯片上,计数使能被简单地标记为 *CTEN*(或者一些诸如 *G* 之类的其他名称),终端计数(*TC*)和一些 IC 计数器上的异步(行波)时钟输出(*RCO*)比较相似。

图 8.40 给出了两个十进制计数器的级联。计数器 1 的终端计数(*TC*)输出连接到计数器 2的计数使能(*CTEN*)输入上。计数器 2 被它的 *CTEN* 输入上的低电平禁止,直至计数器 1 到达它的最后或者终端状态,并且其终端计数输出变为高电平。这个高电平现在使得计数器 2 工作,结果当计数器 1 到达它的终端计数(*CLK*10)之后的第一个时钟脉冲到来时,计数器 2 从它的初始状态变到第二个状态。在计数器 1 完成整个第二次循环之后(当计数器 1 第二次到达终端计数值时),使得计数器 2 再次工作并进入到下一个状态。这个时序继续进行。由于它们是十进制计数器,所以计数器 1 在计数器 2 完成它的第一个循环之前,必须经过 10 个完整的循环。换句话说,对应于计数器 1 的每 10 个循环,计数器 2 经过一个循环。因此,计数器 2 将在 100 个时钟脉冲之后完成一个循环,这两个级联计数器的总的模是 10×10 = 100。



计算机小知识

上一次计算机小知识中所提到的时间印章计数器(TSC)是一个64位计数器。如果此计数器(或者任何全模64位计数器)在100 MHz的时钟频率下工作,观察起来会很有趣,它将耗费5849年的时间来经过它所有的状态,并到达它的终端计数。与此形成对照的是,32位全模计数器在100 MHz的时钟频率下工作时,只需43秒左右的时间就可以经过所有的状态。

当将其视为分频器时,图8.40中的电路就会把输入时钟频率进行100分频。级联计数器常常用来对高频率时钟信号进行分频,从而得到精确度很高的脉冲频率。用于此目的的级联计数器配置有时候称为递减计数链。例如,假设有一个1 MHz的基本时钟频率,想要得到100 kHz、10 kHz、1 kHz的频率,可以使用一系列级联的十进制计数器。如果1 MHz信号被一分为十,输出就是100 kHz。如果100 kHz信号再被一分为十,输出就是10 kHz。再次除以10将会到达1 kHz的频率。这个递减计数链的一般实现方法如图8.41所示。

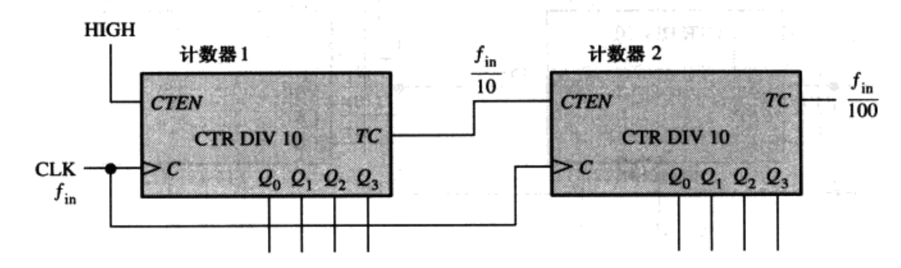


图8.40 使用两个级联十进制计数器的模100计数器

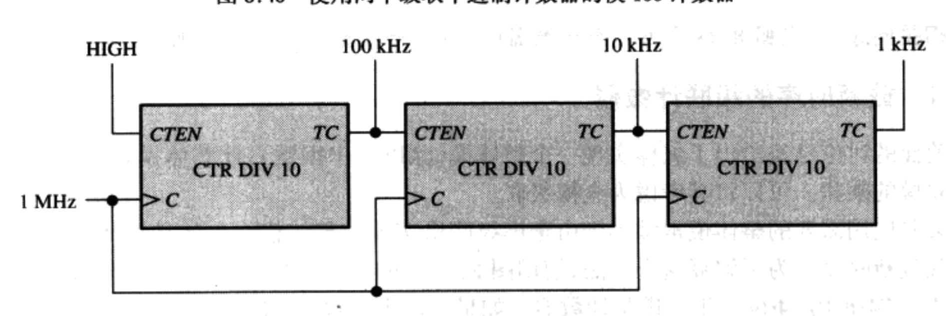


图8.41 3个级联十进制计数器构成1000分频,具有中间10分频和100分频输出

例8.7 确定如图8.42所示的两个级联计数器配置的整体模。

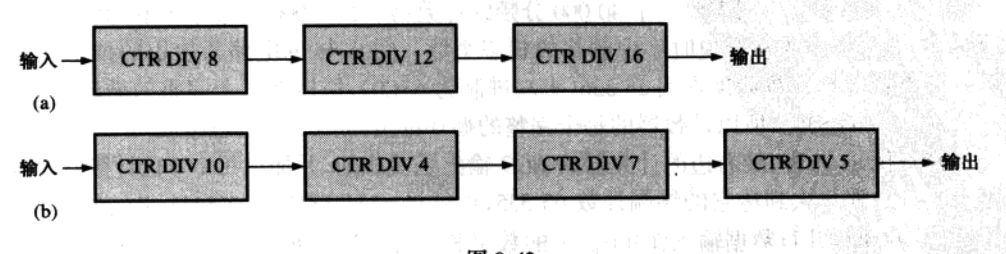


图8.42

解:在图 8.42(a)中,3 计数器配置的整体模是

$$8 \times 12 \times 16 = \mathbf{1536}$$

在图 8.42(b)中,4 计数器配置的整体模是

$$10 \times 4 \times 7 \times 5 = \mathbf{1400}$$

相关问题:把一个时钟频率除以 100 000,需要多少个级联十进制计数器?

例 8.8 使用 74F162 从一个 1 MHz 时钟中得到 10 kHz 的波形。给出逻辑图。

解:为了从 1 MHz 时钟中得到 10 kHz,需要除法因数 100。两个 74F162 计数器必须依照图 8.43所示的那样进行级联。左边的计数器每 10 个时钟脉冲产生一个 *TC* 脉冲。右边的计数器每 100 个时钟脉冲产生一个 *TC* 脉冲。

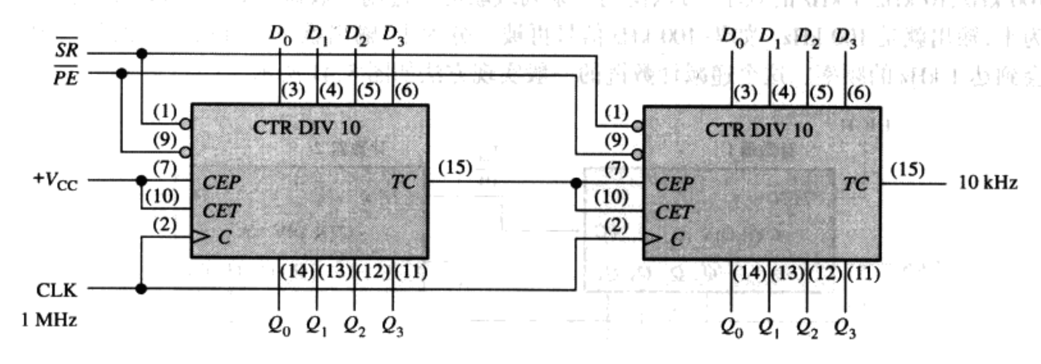


图 8.43 使用两个 74F162 十进制计数器的 100 分频计数器

相关问题:确定图 8.43 中第二个计数器(右边的计数器)的 Q_0 输出上波形的频率。

8.5.1 截断时序的级联计数器

前面的讨论已经给出了怎样实现一个整体模(除以一个因数),此整体模是所有级联计数器独立模的乘积。可以将其考虑为全模级联。

某些应用需要的整体模常常小于由全模级联所实现的整体模。也就是说,级联计数器必须实现截断时序。为了解释这个方法,使用图 8.44 中的级联计数器配置。这个特殊的电路使用了四个 74HC161 4 位同步二进制计数器。如果这四个计数器(总共 16 位)以全模方式级联,模应当是

$$2^{16} = 65\,536$$

假设某个特定的应用需要一个 40 000 分频的计数器(模 40 000)。65 536 和 40 000 的差值为 25 536,也就是必须从全模时序中删除的状态个数。图 8.44 的电路所使用的技术就是,每循环一次都将级联计数器预置到 25 536(十六进制为 63C0),所以在每个完整的循环中它将从 25 536 计数到 65 535。所以计数器的每个完整的循环都由 40 000 个状态组成。

注意在图 8.44 中,最右边的计数器的 *RCO* 输出是反相的,并加在每个 4 位计数器的 $\overline{LOAD}$ 输入上。计数器每次到达它的终端计数 65 535,也就是 $1111\ 1111\ 1111\ 1111_2$ 时,*RCO* 就会变为高电平,并使得并行数据输入($63C0_{16}$)上的数字和时钟脉冲同步进入计数器中。因此,对于每 40 000 个时钟脉冲,最右边的 4 位计数器只有一个 *RCO* 脉冲输出。

使用这种技术，通过将计数器在每次循环中都同步置入合适的初始状态，就可以实现任何模的计数器。

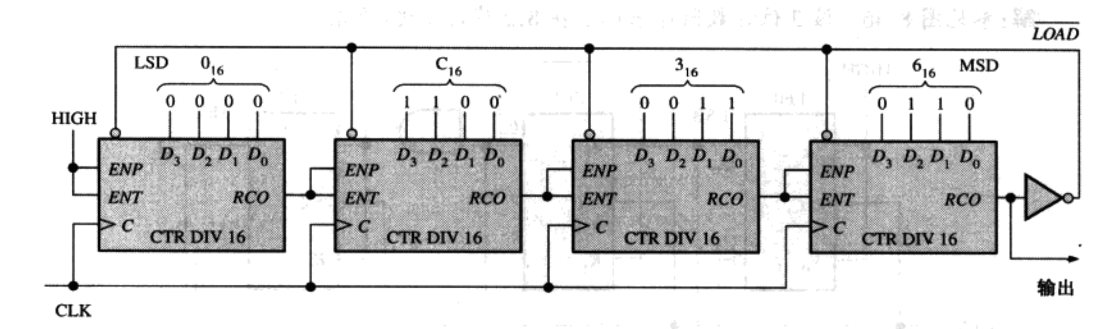


图 8.44　使用 74HC161 4 位二进制计数器的 40 000 分频计数器。注意每个并行的数据输入都以二进制顺序给出(最右边的位 D_0 是每个计数器中的最低有效位)

8.6　计数器译码

在许多应用中，需要对计数器的一些状态或者全部状态译码。计数器的译码涉及译码器或者逻辑门的使用，这些元件用来确定计数器在什么时序时处于什么样的二进制状态。例如，前面所讨论的终端计数函数就是计数器时序中的一个单译码状态(最后的状态)。

学完本节以后，应当能够

- 对于计数器时序中任意给定的状态，实现译码逻辑
- 解释假信号为什么发生在计数器译码逻辑中
- 使用选通方法来消除译码假信号

假设希望译码一个 3 位二进制计数器的状态 6(110)。当 $Q_2=1$、$Q_1=1$ 和 $Q_0=0$ 时，高电平就会出现在译码门的输出上，表示此计数器处于状态 6。实现方法如图 8.45 所示。这称为高电平有效译码。用与非门取代与门，以提供低电平有效译码。

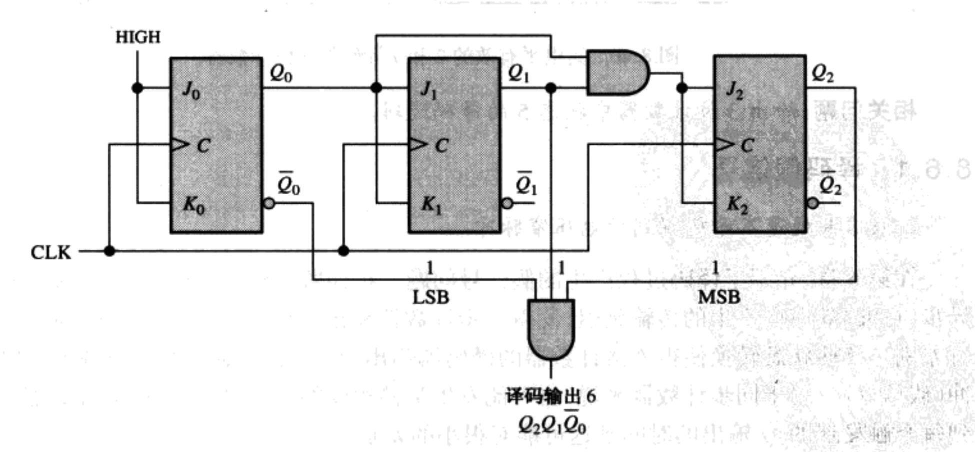


图 8.45　状态 6(110)的译码

例 8.9 实现一个3位同步计数器二进制状态2和二进制状态7的译码。给出整个计数器时序图和译码门的输出波形。二进制 $2=\overline{Q}_2Q_1\overline{Q}_0$,二进制 $7=Q_2Q_1Q_0$。

解: 参见图8.46。该3位计数器原来已经在8.2节讨论过(见图8.14)。

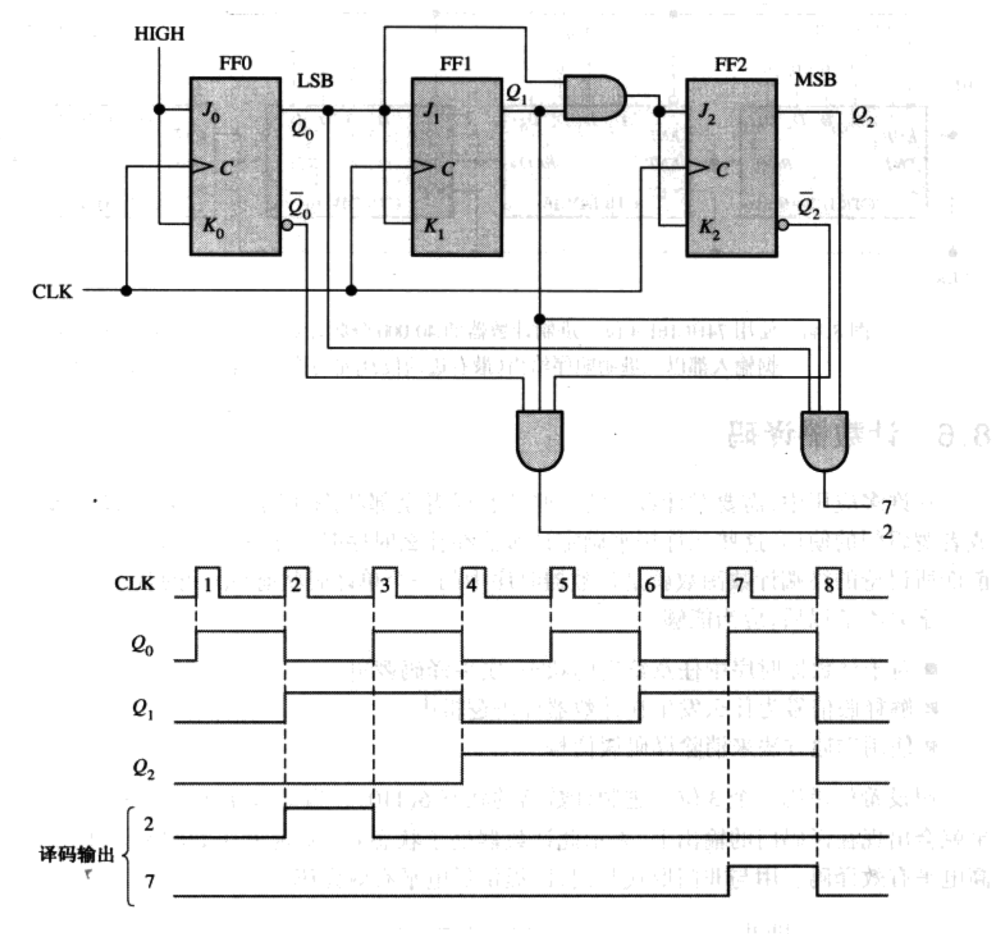


图 8.46 高电平有效的2和7译码的3位计数器

相关问题: 给出3位计数器中状态5的译码逻辑。

8.6.1 译码假信号

◇ 假信号就是不希望得到的电压窄脉冲。

在第6章,介绍了译码过程产生的假信号问题。正如所看到的那样,由于异步计数器中的异步(行波)效应所产生的传输延迟,使得异步计数器输出变化而产生的状态转换在时间上稍有差别。这些状态转换使得连接计数器的译码器输出上,产生短期的不希望得到的电压窄脉冲(假信号)。对于同步计数器来说,也可能发生某种程度的假信号问题,因为计数器中从时钟到每个触发器的 Q 输出的时间延迟可能有很小的差别。

图8.47给出了一个基本异步 BCD 十进制计数器,它连接到一个 BCD - 十进制译码器上。

为了观察在此情况下会发生什么,我们来看时序图,其中也考虑了传输延迟,如图 8.48 所示。注意这些延迟产生了短期的错误状态。每个临界转换处的错误二进制状态值表示在图上。在译码器输出上可以看到假信号结果。

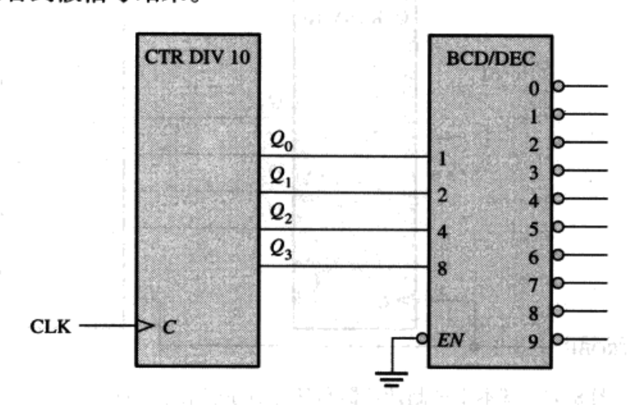


图 8.47 基本十进制(BCD)计数器和译码器

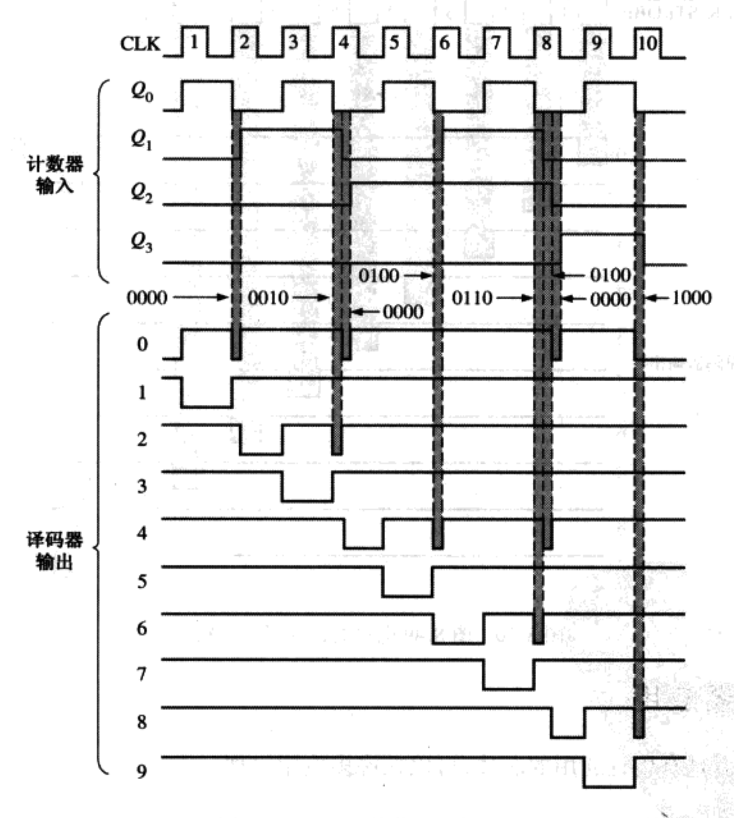


图 8.48 图 8.47 中译码器的具有假信号的输出。为了解释方便,图中将假信号宽度夸大了,实际上假信号通常只有几纳秒宽

一种消除假信号的方法是在假信号消失以后才能够译码输出。这种方法称为选通(strobe),

可以使用高电平有效时钟(对计数器而言)的低电平来使译码器工作,如图 8.49 所示。最后改善的时序图如图 8.50 所示。

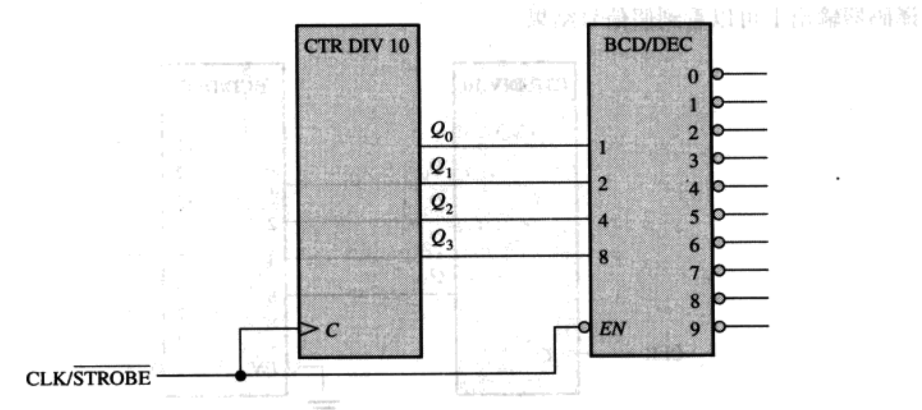


图 8.49 基本十进制计数器和具有消除假信号的选通译码器

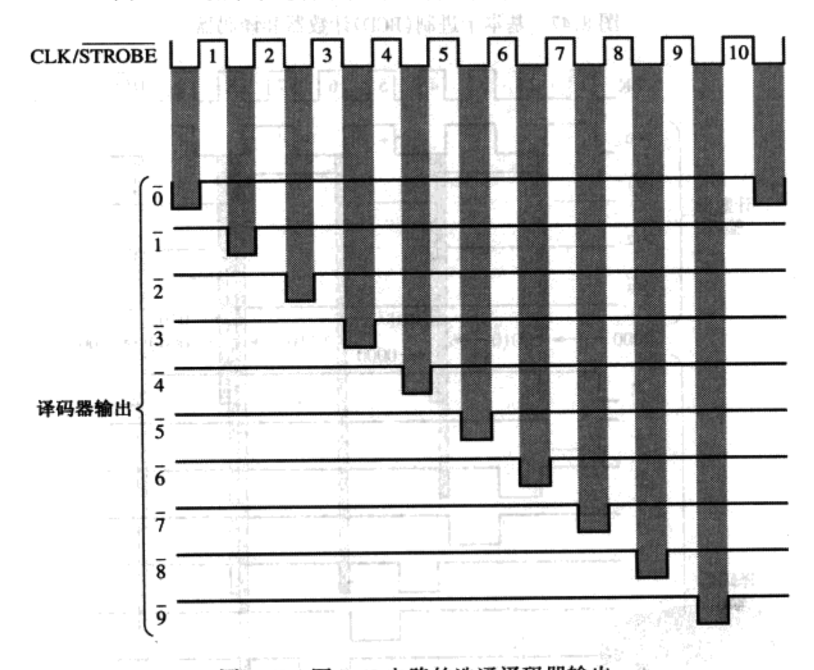


图 8.50 图 8.49 电路的选通译码器输出

8.7 计数器应用

数字计数器是有用且通用的芯片,可以在许多应用出现。在本节中,介绍了一些具有代表性的计数器应用。

学完本节以后,应当能够

- 描述计数器怎样应用于基本数字时钟系统中
- 解释怎样实现 60 分频计数器及它怎样应用于数字时钟

- 解释怎样实现小时计数器
- 讨论计数器在停车控制系统中的应用
- 描述计数器怎样应用在并行数据到串行数据的转换过程中

8.7.1 数字时钟

计数器应用的一个常见例子是计时系统。图 8.51 是显示秒、分、小时的数字时钟的简化逻辑图。首先,60 Hz 的正弦交流电压变换为 60 Hz 的脉冲波形,并且由一个 60 分频的计数器分频为 1 Hz 的脉冲波形,这个计数器由一个 10 分频计数器紧接着一个 6 分频的计数器组成。秒和分计数也是由 60 分频的计数器产生,其细节如图 8.52 所示。这些计数器从 0 计数到 59 然后再循环回到 0;同步十进制计数器用在了这个特殊的实现方法中。注意 6 分频的部分由一个具有截断时序的十进制计数器组成,截断时序通过译码器计数 6 异步清零计数器来实现。终端计数 59 也被译码,以使计数器链中的下一个计数器工作。

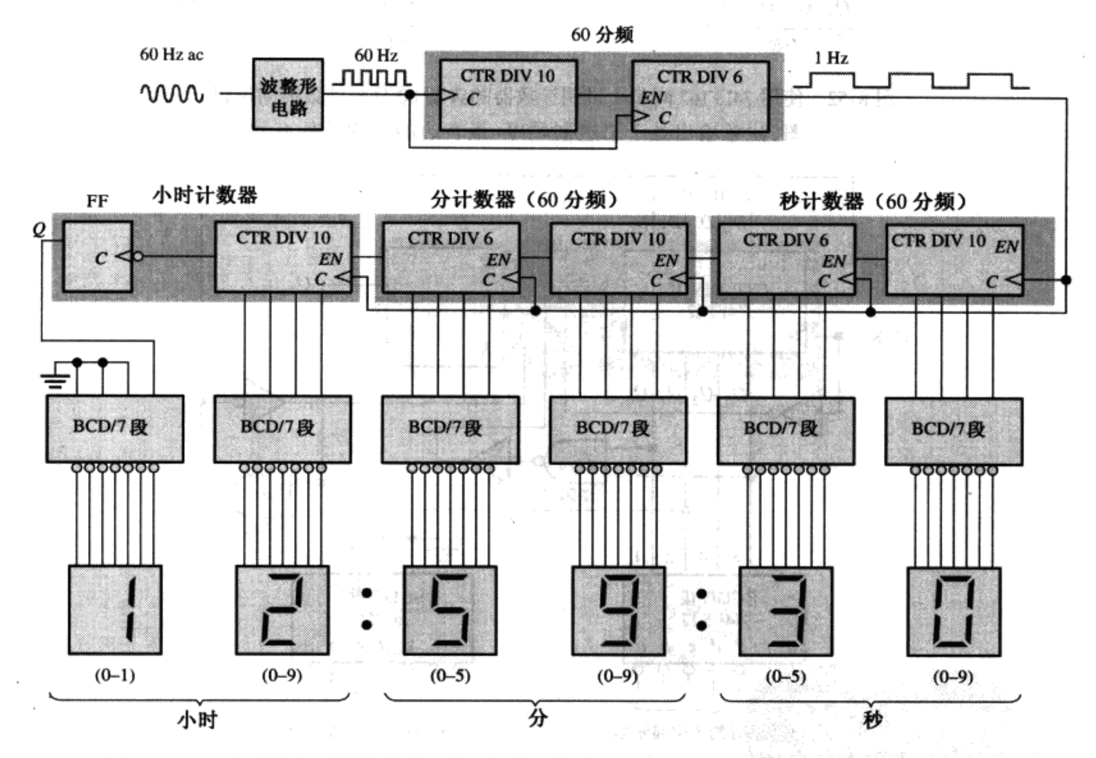


图 8.51 12 小时数字时钟的简化逻辑图,使用特定芯片的逻辑细节如图 8.52 和图 8.53 所示

小时计数器由一个十进制计数器和一个触发器组成,如图 8.53 所示。考虑到初始时两个十进制计数器和触发器都处在复位状态,并且译码 12 门和译码 9 门输出都是高电平。十进制计数器顺序经过从 0 到 9 的所有状态,并且在从 9 再循环回到 0 的时钟脉冲到来时,触发器进入置位状态($J=1, K=0$)。这样就可点亮十位小时显示器上的 1。总计数现在为 10(十进制计数器处于 0 状态和触发器为置位状态)。

接下来,总计数顺序计到 11 然后再到 12。在状态 12 时,十进制计数器的 Q_2 输出为高电

平,触发器仍然为置位,因此译码 12 门输出为低电平。这就使十进制计数器的$\overline{PE}$输入有效。在下一个时钟脉冲到来时,十进制计数器由数据输入预置到状态 1,而触发器处于复位($J=0$,$K=1$)。正如所看到的那样,这个逻辑总是使得计数器从 12 再循环回到 1 而不是 0。

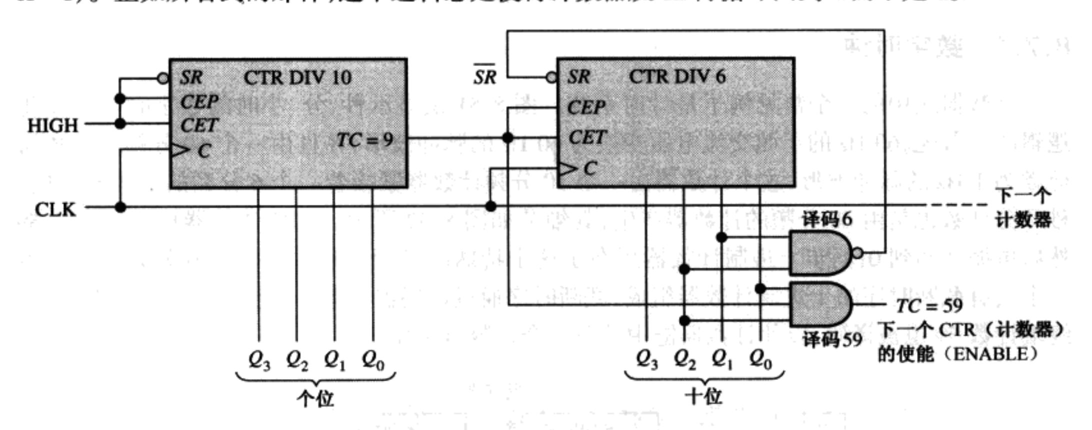


图 8.52 使用 74LF162 同步十进制计数器的典型 60 分频计数器的逻辑图,注意输出处于二进制顺序(最右边的位为最低有效位)

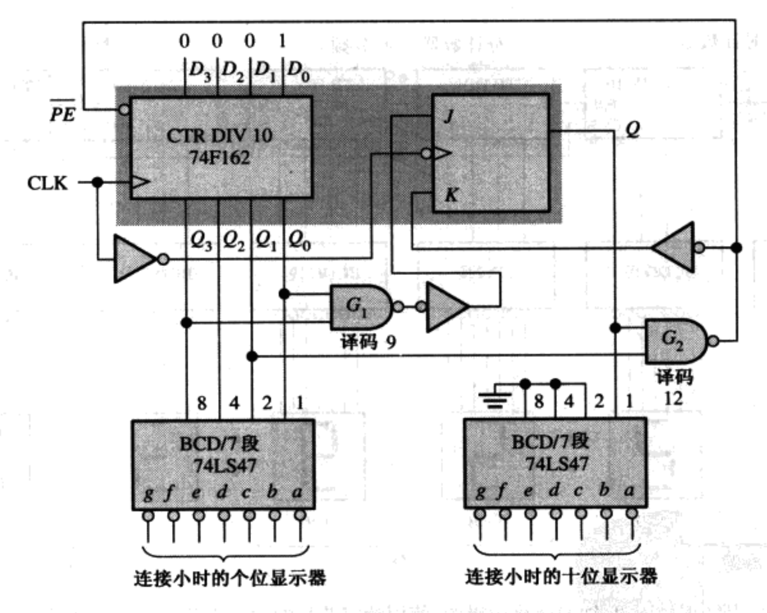


图 8.53 小时计数器和译码器的逻辑图,注意在计数器输入和输出上,最右边的位是最低有效位

8.7.2 停车控制

这个计数器例子阐述了使用加/减计数器去解决日常问题。这个问题是设计一种检测手段,检测在拥有 100 个车位的停车库中的可用车位,并且通过点亮显示信号以提供停满条件的指示,同时降下入口处的门栏杆。

解决此问题的系统由以下几部分构成:(1)车库入口和出口处的光电传感器;(2)一个加/减计数器和相关电路;(3)接口电路,它使用计数器输出去打开或者关闭停满信号,并且降下或者举起入口处的门栏杆。这个系统的总体框图如图 8.54 所示。

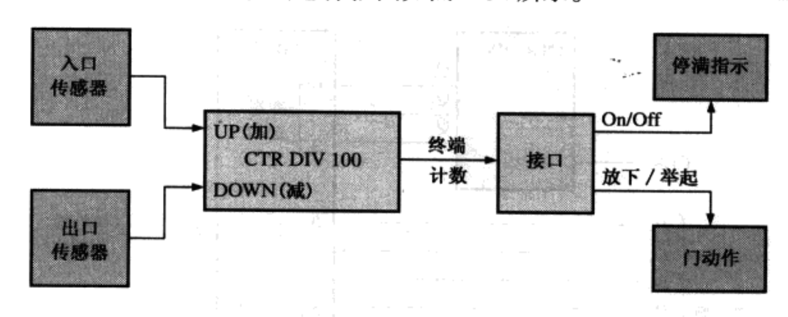


图 8.54 停车库的功能框图

加/减计数器的逻辑图如图 8.55 所示。它由两个级联 74HC190 加/减十进制计数器组成。下面一段描述它的运算。

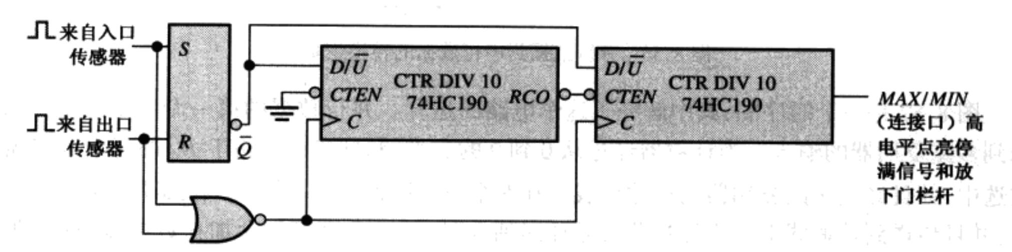


图 8.55 用于停车控制的模 100 加/减计数器的逻辑图

◇ 计数器递增就是将它的计数值依次增加 1。

使用并行数据输入把计数器初始预置为 0,这里没有给出。每个进入车库的汽车都会切断一个光束,从而使得传感器产生一个电子脉冲。这个正脉冲的上升沿把 S-R 锁存器置位。锁存器 $\overline{Q}$ 输出上的低电平使得计数器进入加模式。同样,此传感器脉冲经过或非门,并在它后沿的低电平到高电平的转换时计数器加 1。每次汽车进入车库时,计数器就增加 1(递增)。当第一百辆车进入车库时,计数器就到达它的最后一个状态(100_{10})。*MAX/MIN* 输出变为高电平,并使得接口电路工作(没有细节),点亮停满指示灯并降下门栏杆以阻止其他车辆进入。

当汽车开出车库时,光电传感器就会产生一个正脉冲,复位 S-R 锁存器并将计数器置于减模式。时钟的后沿将计数减 1(递减)。如果车库停满而车辆离开,计数器的 *MAX/MIN* 输出就会变为低电平,并关闭停满指示灯,同时抬起门栏杆。

◇ 计数器递减就是将它的计数值依次减去 1。

8.7.3 并行数据到串行数据的转换(多路复用)

在第 6 章介绍了使用多路复用和多路分配技术的数据传送的简化例子。本质上,多路复用器输入上的并行数据位被转换为单传输线上的串行数据位。同时出现在并行线上的一个位组称为并行数据。以时间顺序出现在单线上的位组称为串行数据。

并行到串行的转换通常使用计数器来实现,计数器为数据选择器/多路复用器的数据选择输入提供一个二进制序列,如图 8.56 所示。模 8 计数器的 Q 输出连接到 8 位多路复用器的数据选择输入端。

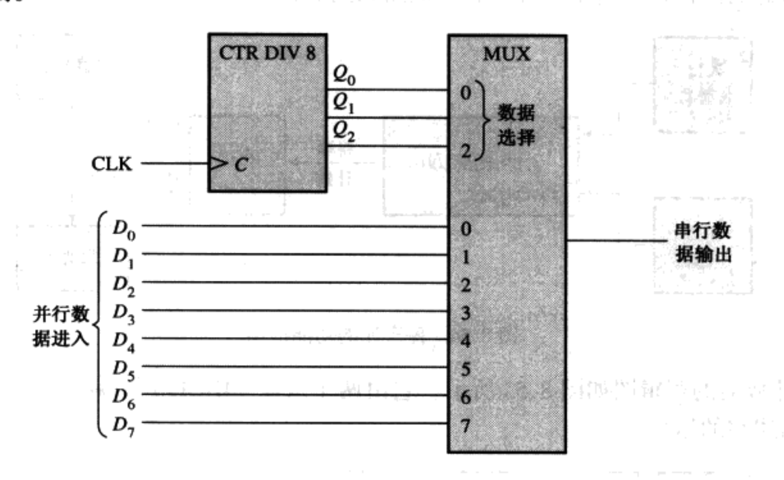


图 8.56 并行数据到串行数据的转换逻辑

图 8.57 是一个时序图,其中解释了这个电路的运算。并行数据的第一个字节(8 位一组)加到多路复用器的输入。当计数器经过从 0 到 9 的二进制时序时,开始于 D_0 的每个位按顺序被选中,并且经过多路复用器到达输出线。在 8 个时钟脉冲之后,数据字节已经转换为串行格式,并且传送到传输线上。当计数器再循环回到 0 时,下一个字节就会加在数据输入端,随着计数器循环经过它的 8 个状态,这个字节又被顺序转换为串行格式。这个过程会重复继续下去,每个并行字节都被转换为串行字节。



计算机小知识

计算机含有一个内部计数器,可以对不同的音频和音调长短进行编程,由此产生了“音乐”。为了选择特殊的音调,编程指令就会选择一个除数传送到计数器。这个除数设置计数器去除以外部时钟的基率,从而产生声频音调。音调的长短可以由编程指令来设置;因此,使用基本计数器通过控制频率和音调长短来产生美妙的音乐。

8.8 关联标注的逻辑符号

到目前为止,我们已经大致介绍了 ANSI/IEEE 标准 91-1984 中指定的关联标注逻辑符号。在许多情况下,新符号不会在很大程度上偏离传统符号。但是,对于某些芯片,确实发生了值得注意的偏离惯用符号的情况,其中包括计数器和其他更复杂的芯片。尽管将在本书中继续使用更加传统且熟悉的符号,但是仍然提供了对相关表示的逻辑符号的简单介绍。本节使用一个特殊的 IC 计数器作为一个示例。

学完本节以后,应当能够

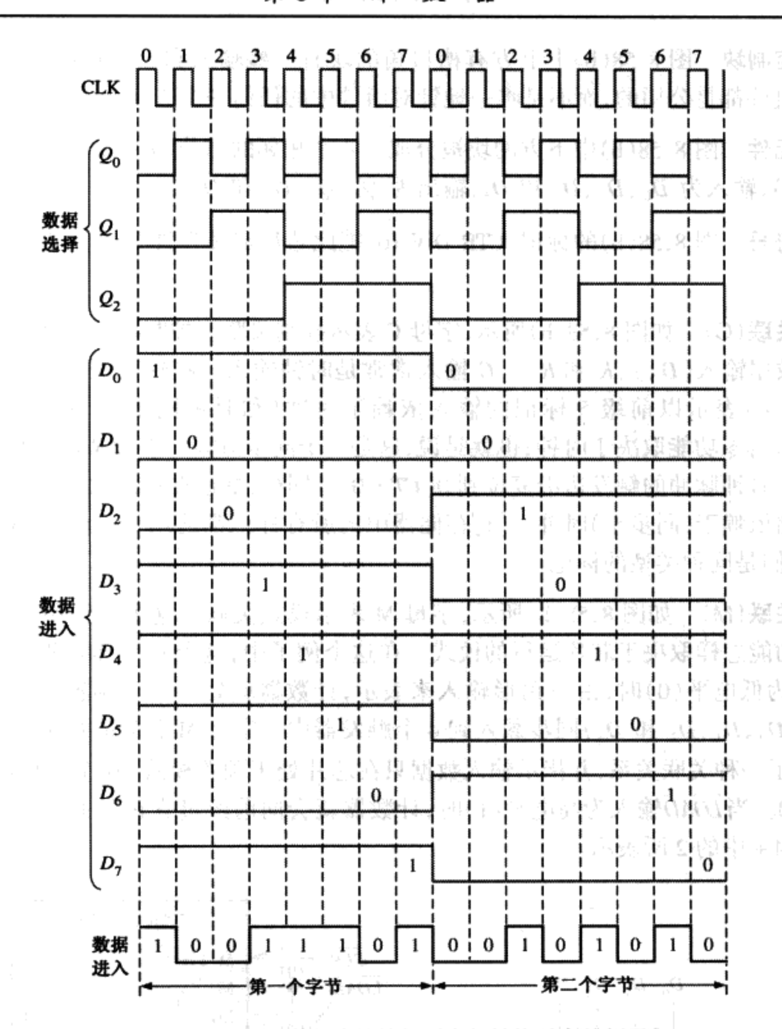


图 8.57 图 8.56 中电路的并行到串行转换时序的例子

- 解释包括关联标注的逻辑符号
- 识别计数器符号的共用框图和独立元件
- 解释限制符号
- 讨论控制关联
- 讨论模式关联
- 讨论与关联

关联标注是 ANSI/IEEE 标准的基础。关联标注和逻辑符号联合在一起，用以指定输入和输出的关系，使得某个给定芯片的逻辑运算可以完全从它的逻辑符号来确定，而不用事先知道其内部结构的细节，也不需要详细的逻辑图来作为参考。介绍具有关联标注的特定逻辑符号，是为了帮助解释将来可能遇到的其他此类符号。

我们用 74HC163 4 位同步二进制计数器进行解释。为了进行比较，图 8.58 给出了一个传统的块图符号和具有关联标注的 ANSI/IEEE 符号。此符号和关联标注的基本描述如下。

共用控制块　图 8.58(b)中上方有槽口角的块有一些输入和一个输出,它们对于该芯片中所有的组件都是公用的,而不是唯一地针对组件中的任何一个。

独立元件　图 8.58(b)中下方的块被分成了 4 个相邻的部分,表示计数器中 4 个存储元件(D 触发器),输入为 D_0、D_1、D_2 和 D_3,输出为 Q_0、Q_1、Q_2 和 Q_3。

限制符号　图 8.58(b)的标记"CTR DIV 16"把此芯片定义为具有 16 个状态(DIV 16)的计数器(CTR)。

控制关联(C)　如图 8.58(b)所示,字母 C 表示控制关联。控制输入常常使能或者禁止存储元件的数据输入(D、J、K 和 R)。C 输入常常是时钟输入。在这个例子中,跟随 C 的数字(C5/2,3,4 +)表示以前缀 5 标记的输入依赖于时钟(和时钟同步)。例如,$\overline{CLR}$ 输入上的 5CT = 0表示清零功能取决于时钟;也就是说,这是一个同步清零。当$\overline{CLR}$输入为低电平(0)时,计数器就在时钟脉冲的触发边沿复位到 0(CT = 0)。同样,标记在存储元件[1]上的 5D 表示这个数据存储依赖于(同步于)时钟。[1]存储器中的所有标记都适用于下面的[2]、[4]和[8]元件,因而它们是同种类型的标记。

模式关联(M)　如图 8.58(b)所示,字母 M 表示模式关联。这个标记用来表示不同输入或输出的功能怎样取决于芯片运行的模式。在这个例子中,这个芯片具有两个运行模式。当$\overline{LOAD}$输入为低电平(0)时,由三角形输入来表示,计数器就处于一个预置的模式(M1),其中输入数据(D_0、D_1、D_2 和 D_3)同步载入到 4 个触发器中。字母 M 后面的数字 1 及标记 1 中的 1,5D 给出了一种关联关系,并指示输入数据只在芯片处于预置模式(M1)时才会存储起来,此时$\overline{LOAD}$ = 0。当$\overline{LOAD}$输入为高电平(1)时,计数器就会向前经过它正常的二进制时序,由 M2 和 C5/2,3,4 + 中的 2 所表示。

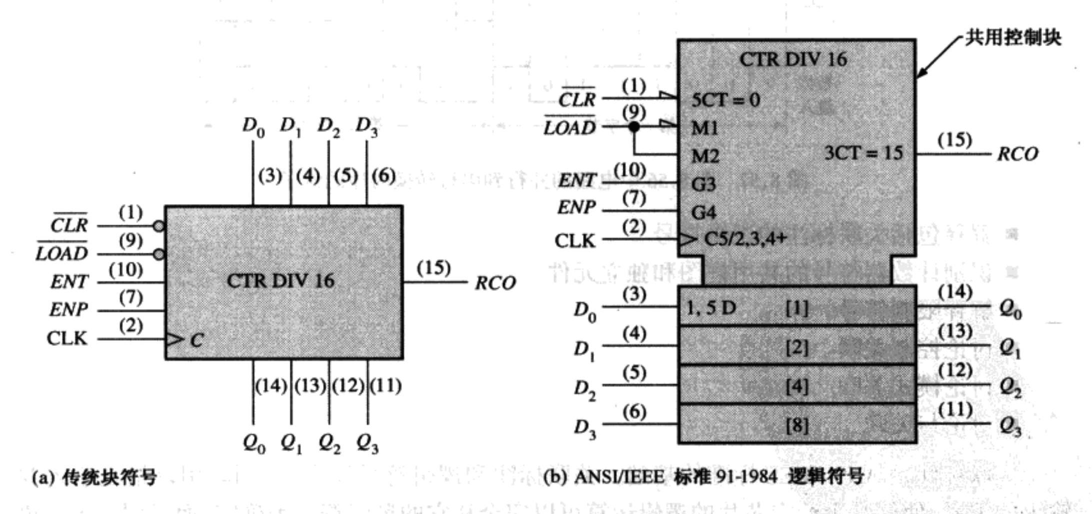


图 8.58　74HC163 4 位同步计数器

与关联(G)　如图 8.58(b)所示,字母 G 表示与关联,说明由 G 与其后面的一个数字所命名的输入与任何其他标记中具有相同数字前缀的输入和输出执行与运算。在这个特殊的例子中,ENT 输入上的 G3 和 RCO 输出上的 3CT = 15 相关,由数字 3 来指示,并且其关系是与关联,

由字母 G 来指示。这说明如果 RCO 输出是高电平，ENT 必须是高电平（输入上没有三角形）而计数必须为 15（$CT=15$）。

同样，标记 C5/2,3,4+ 中的数字 2、3 和 4 指出当 $\overline{LOAD}=1$ 时，如图所示为模式关联标记 M2，计数器顺序经过它的状态；当 $ENT=1$ 和 $ENP=1$ 时，如图所示为与关联标记 G3 和 G4。“+”指出当这些情况存在时，计数器顺序递增 1。

8.9 数字系统应用

交通灯控制系统的讲解从第 6 章开始，并在第 7 章继续进行讨论，本章将完成这个实例的学习。我们在第 6 章介绍了组合逻辑电路，并在第 7 章介绍了定时电路。

本章介绍时序电路，并且把所有的框图连接起来组成完整的交通控制系统。整个系统的框图如图 8.59 所示。

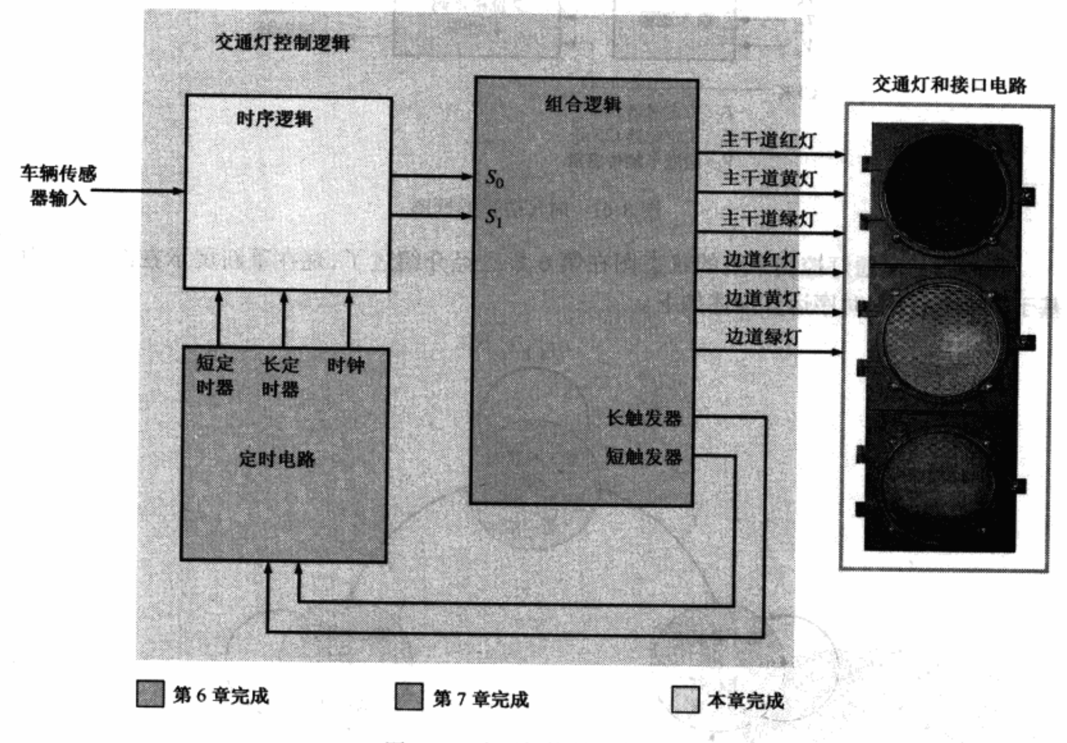


图 8.59 交通灯控制系统框图

8.9.1 时序逻辑要求

时序逻辑电路控制交通灯基于输入的时序，这些输入来自定时电路和车辆传感器。时序逻辑电路产生系统四种状态的 2 位格雷码时序，如图 8.60 所示。

框图 时序逻辑由一个 2 位格雷码计数器和相关的输入逻辑组成，如图 8.61 所示。

计数器产生四种状态的一个时序。从一个状态到下一个状态的转换由 4 秒定时器、25 秒定时器和车辆传感器确定。计数器的时钟为 10 kHz 信号，由定时电路中的振荡器产生。

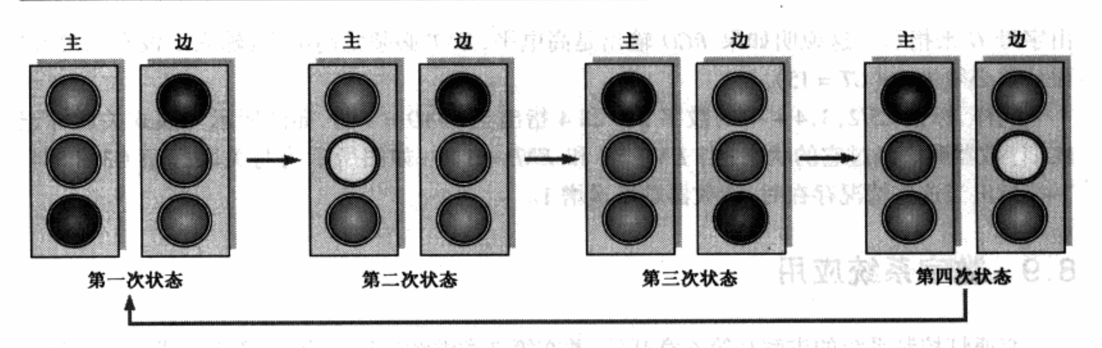


图 8.60　交通灯状态的时序

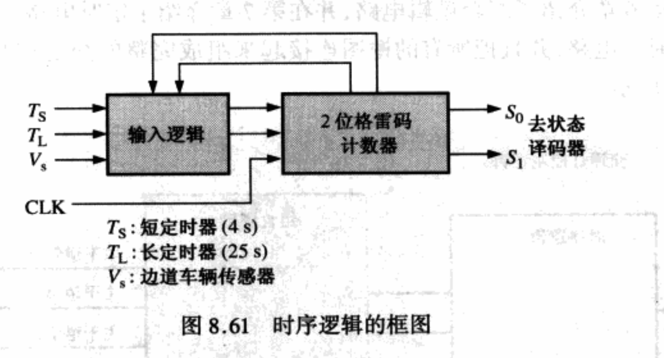


图 8.61　时序逻辑的框图

状态图　交通灯控制系统的状态图在第 6 章已经介绍过了，现在重新展示在图 8.62 中。基于这个状态图，时序运算描述如下。

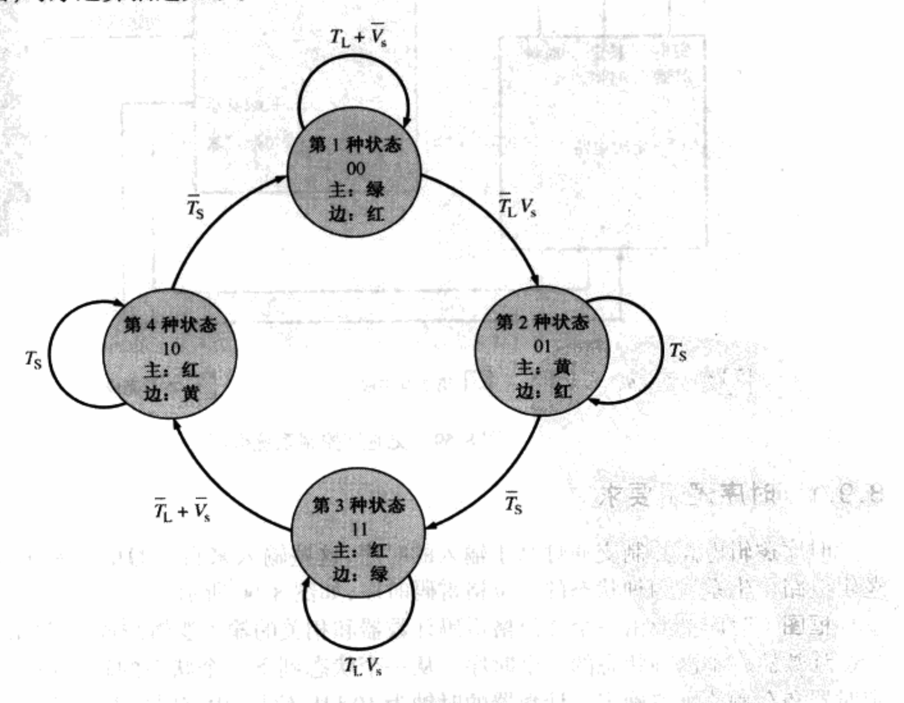


图 8.62　交通灯控制系统的状态图

第一个状态:该状态的格雷码是00。主干道交通灯是绿灯而边道灯为红灯。当长定时器打开或者只要边道上没有汽车,这表示为 $T_L+\bar{V}_s$,这时系统就会保持在这个状态至少25秒。当长定时器关闭或边道上有汽车,表示为($\bar{T}_L V_s$),这时系统进入下一个状态。

第二个状态:该状态的格雷码是01。主干道为黄灯(警告)而边道为红灯。当短定时器打开(T_s)时,系统就会保持在这个状态4秒,而当短定时器关闭($\bar{T}_s$)时,系统就会进入下一个状态。

第三个状态:此状态的格雷码是11。主干道为红灯而边道为绿灯。当长定时器打开并且边道有车辆,表示为($T_L V_s$),这时系统就会保持在这个状态。当长定时器关闭或者边道上没有车辆,表示为 $\bar{T}_L+\bar{V}_s$,这时系统进入下一个状态。

第四个状态:状态的格雷码是10。主干道是红灯而边道是黄灯。当短定计时器打开(T_S)时,该系统将保持在这个状态4秒,而当短定时器关闭($\bar{T}_S$)时,系统就会返回第一个状态。

实现时序逻辑 图8.63中的图给出了用来实现格雷码计数器的两个D触发器。D触发器的输入由来自输入逻辑的输出提供,计数器的时钟频率为10 kHz,由振荡器提供。输入逻辑有五个变量:Q_0, Q_1, T_L, T_S, V_s。

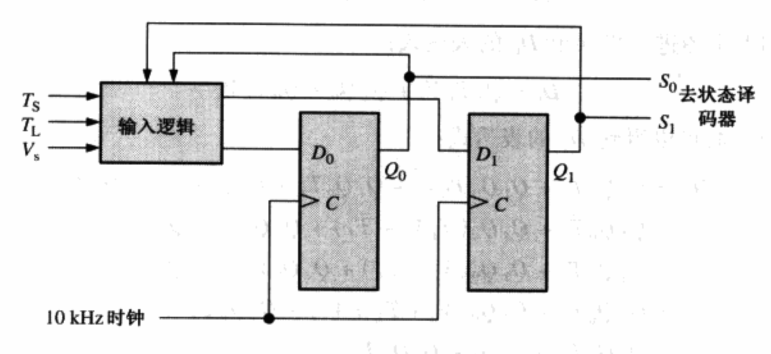


图8.63 时序逻辑图

D触发器的转换表如图8.13所示。从状态图中可以得到如表8.14的次态表。每个当前状态/次态的组合的输入情况在表中列出。

根据表8.13和表8.14,可以确定每个触发器进入1状态所需要的逻辑条件。例如,当前状态是00时,Q_0 从0变化到1。输入条件 $\overline{T_L} V_s$ 在表8.13的第二行给出。D_0 必须为1以使 Q_0 变为1,或在下一个时钟脉冲到来时保持为1。对于 D_0 为1的情况,可以根据表8.14写出逻辑表达式:

表8.13 D触发器转换表

输入转换			触发器输入
Q_N		Q_{N+1}	D
0	⟶	0	0
0	⟶	1	1
1	⟶	0	0
1	⟶	1	1

表 8.14　时序逻辑转换的次态表

当前状态		次　态			FF 触发器	
Q_1	Q_0	Q_1	Q_0	输入条件	D_1	D_0
0	0	0	0	$\overline{T}_L + V_s$	0	0
0	0	0	1	$T_L\overline{V}_s$	0	1
0	1	0	1	$\overline{T}_S$	0	1
0	1	1	1	T_S	1	1
1	1	1	1	$\overline{T}_L\overline{V}_s$	1	1
1	1	1	0	$T_L + V_s$	1	0
1	0	1	0	$\overline{T}_S$	1	0
1	0	0	0	T_S	0	0

$$D_0 = \overline{Q}_1\overline{Q}_0\overline{T}_L V_s + \overline{Q}_1 Q_0 T_S + \overline{Q}_1 Q_0\overline{T}_S + Q_1 Q_0 T_L V_s$$
$$= \overline{Q}_1\overline{Q}_0\overline{T}_L V_s + \overline{Q}_1 Q_0 + Q_1 Q_0 T_L V_s$$

可以使用卡诺图进一步导出 D_0 的表达式：

$$D_0 = \overline{Q}_1\overline{T}_L V_s + \overline{Q}_1 Q_0 + Q_0 T_L V_s$$

同样，根据表 8.14，可以得到 D_1 的表达式：

$$D_1 = \overline{Q}_1\overline{Q}_0\overline{T}_S + Q_1 Q_0 T_L V_s + Q_1 Q_0\overline{T}_L + Q_1 Q_0\overline{V}_s + Q_1\overline{Q}_0 T_S$$
$$= \overline{Q}_1 Q_0\overline{T}_S + Q_1 Q_0(T_L V_s + \overline{T}_L) + Q_1 Q_0\overline{V}_s + Q_1\overline{Q}_0 T_S$$
$$= \overline{Q}_1 Q_0\overline{T}_S + Q_1 Q_0(V_s + \overline{T}_L) + Q_1 Q_0\overline{V}_s + Q_1\overline{Q}_0 T_S$$
$$= \overline{Q}_1 Q_0\overline{T}_S + Q_1 Q_0(V_s + \overline{T}_L + \overline{V}_s) + Q_1\overline{Q}_0 T_S$$
$$= \overline{Q}_1 Q_0\overline{T}_S + Q_1 Q_0 + Q_1\overline{Q}_0 T_S$$

可以使用卡诺图进一步推导 D_1 的表达式：

$$D_1 = Q_0\overline{T}_S + Q_1 T_S$$

实现 D_0 和 D_1 的电路如图 8.64 所示

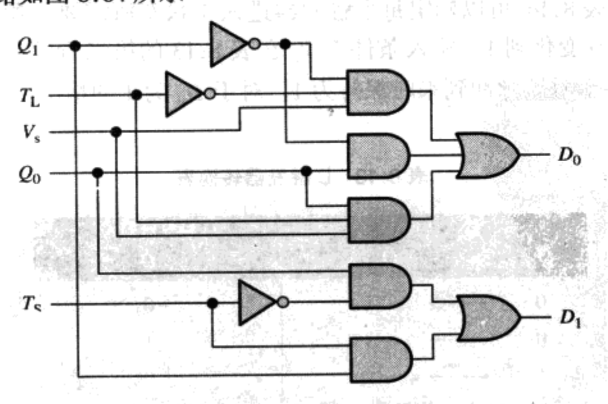


图 8.64　2 位格雷码计数器的输入逻辑

输入逻辑和2位计数器合起来,完整的时序逻辑图如图8.65所示

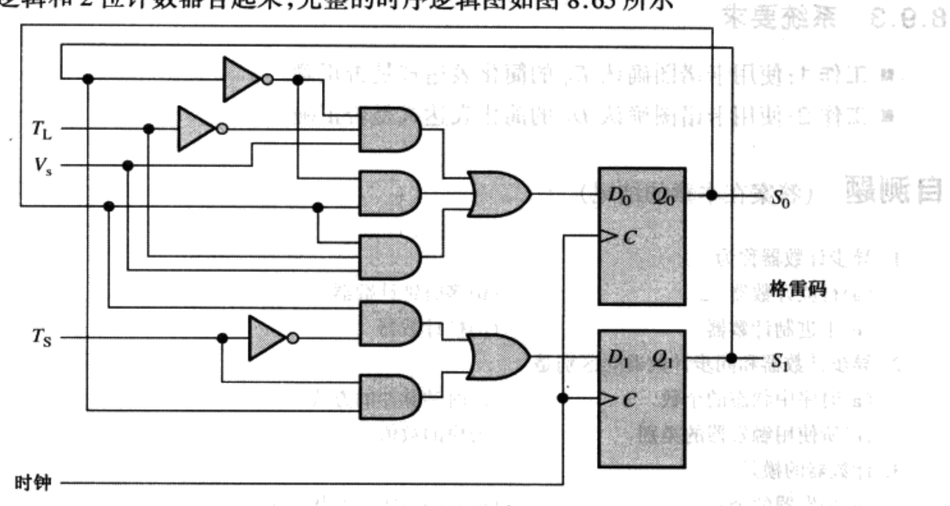


图8.65 时序逻辑

8.9.2 完整的交通灯控制系统

现在有了全部的三个框图(组合逻辑电路、定时器电路和时序逻辑电路),把它们合并起来形成一个完整的系统,如图8.66的框图所示。

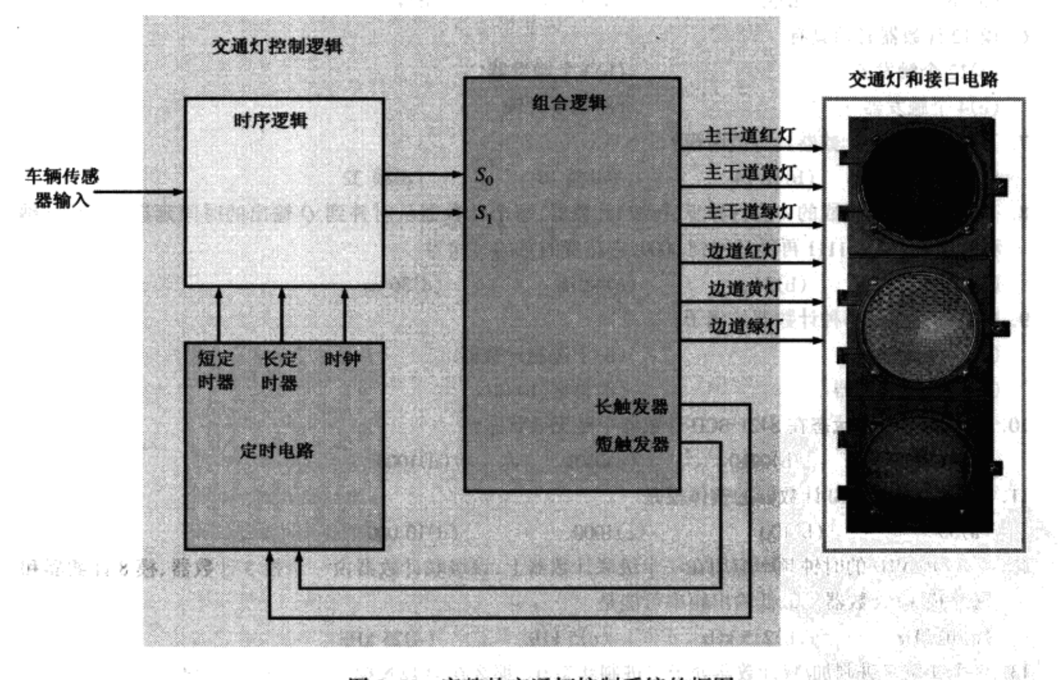


图8.66 完整的交通灯控制系统的框图

接口电路 接口电路是必需的,因为由于电流和电压的需要,逻辑电路不能直接驱动交通灯。有几种提供接口的设计方法,这里不再详述。

8.9.3 系统要求

- 工作 1:使用卡诺图确认 D_0 的简化表达式是否正确。
- 工作 2:使用卡诺图确认 D_1 的简化表达式是否正确。

自测题 (答案在本章的结尾)

1. 异步计数器称为
   (a)行波计数器　(b)多时钟计数器
   (c)十进制计数器　(d)模计数器
2. 异步计数器和同步计数器的区别是
   (a)时序中状态的个数　(b)时钟脉冲的方法
   (c)所使用触发器的类别　(d)模的数值
3. 计数器的模是
   (a)触发器的个数　(b)时序中实际的状态个数
   (c)一秒钟内再循环的次数　(d)状态的最大可能个数
4. 3 位二进制计数器的最大模是
   (a)3　(b)6　(c)8　(d)16
5. 4 位二进制计数器的最大模是
   (a)16　(b)32　(c)8　(d)4
6. 模 12 计数器必须具有
   (a)12 个触发器　(b)3 个触发器
   (c)4 个触发器　(d)同步时钟
7. 下面的哪一个计数器具有截断模?
   (a)模 8　(b)模 14　(c)模 16　(d)模 32
8. 一个由触发器组成的 4 位行异步(行波)计数器,每个计数器从时钟到 Q 输出的时间延迟为 12 ns(纳秒)。计数器从 1111 再循环回到 0000,它花费的总的时间为
   (a)12 ns　(b)24 ns　(c)48 ns　(d)36 ns
9. BCD 计数器是哪种计数器的例子?
   (a)全模计数器　(b)十进制计数器
   (c)截断模计数器　(d)答案(b)和(c)
10. 下面的哪一个状态在 8421 BCD 计数器中是无效状态?
    (a)1100　(b)0010　(c)0101　(d)1000
11. 三个级联的模 10 计数器的整体模是
    (a)30　(b)100　(c)1000　(d)10 000
12. 一个 10 MHz 的时钟频率应用在一个级联计数器上,该级联计数器由一个模 5 计数器、模 8 计数器和两个模 10 计数器。最低输出频率可能是
    (a)10 kHz　(b)2.5 kHz　(c)5 kHz　(d)25 kHz
13. 一个 4 位二进制加/减计数器处于二进制状态 0。那么在 DOWN 模式中的下一个状态是
    (a)0001　(b)1111　(c)1000　(d)1110
14. 模 13 二进制计数器的终端计数是
    (a)0000　(b)1111　(c)1101　(d)1100

15. 下面的哪个 ABEL 等式是指寄存输出？

(a) = A & B;　(b)X = !(& B)　(c)X: - A & B　(d) X- >A & B

习题

8.1节　异步计数器运算

1. 对于如图 8.67 所示的异步计数器，对于 8 个时钟脉冲给出完整的时序图，同时给出时钟 Q_0 和 Q_1 的波形。

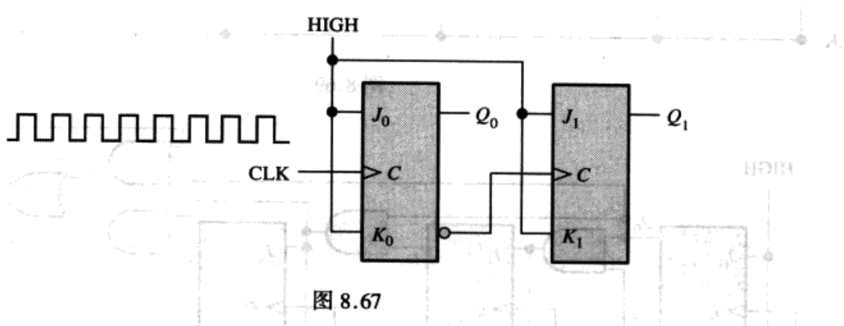


图 8.67

2. 对于图 8.68 中的异步计数器，对于 6 个时钟脉冲给出完整的时序图。给出时钟 Q_0、Q_1 和 Q_2 的波形。

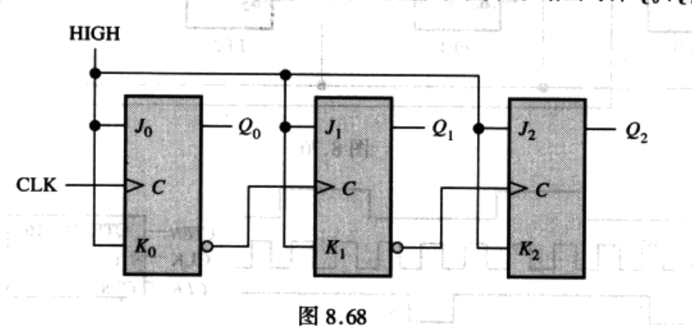


图 8.68

3. 在习题 2 的计数器中，假设每个触发器从时钟的触发边沿到 Q 输出的变化都有一个 8 ns 的传输延迟。在给定的状态下，确定从时钟脉冲到计数器之间的最差状况(最长)延迟时间。指定状态或最差状态延迟发生时的状态。
4. 给出怎样为下面的每一个模连接 74LS93 4 位异步计数器：

(a)9　(b)11　(c)13　(d)14　(e)15

8.2节　同步计数器运算

5. 如果习题 3 中的计数器为同步而不是异步的，那么最长延迟时间应当是多少？
6. 为图 8.69 中的 5 级同步二进制计数器给出完整的时序图。验证 Q 输出的波形表示每个时钟脉冲后的正确二进制数。
7. 通过分析每个时钟脉冲之前的每个触发器的 J 和 K 输入，证明图 8.70 中的十进制计数器经过一个 BCD 序列。解释每种情况下的这些条件怎样使得计数器进入下一个正确的状态。
8. 图 8.71 中的波形应用于计数使能端、清零及时钟输入。给出正确的和这些输入相关的计数器输出波形。清零输入是异步的。
9. BCD 十进制计数器如图 8.72 所示。波形按照所指示的那样应用于时钟和清零输入。确定每个计数器输出(Q_0, Q_1, Q_2, Q_3)的波形。清零是同步的，并且计数器初始时处于二进制 1000 状态。

图 8.69

图 8.70

图 8.71

10. 图 8.73 中的波形应用于一个 74HC163 计数器中。确定 Q 输出和 RCO。输入为 $D_0=1, D_1=1, D_2=0, D_3=1$。

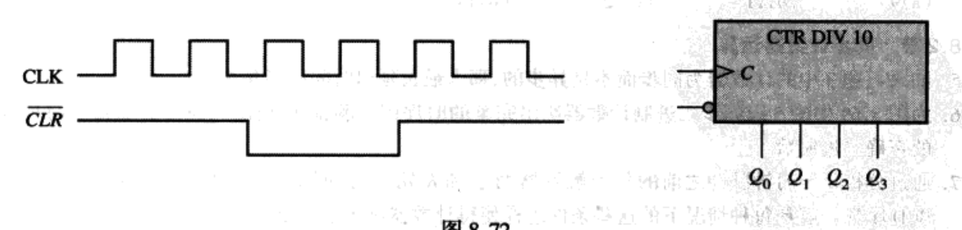


图 8.72

11. 图 8.73 中的波形应用于一个 74F162 计数器中。确定 Q 输出和 TC。输入为 $D_0=1, D_1=0, D_2=0, D_3=1$。

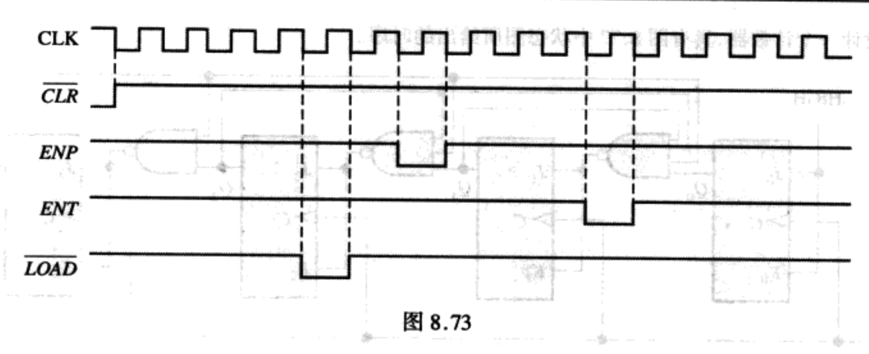


图 8.73

8.3节 加/减同步计数器

12. 为经过下面时序的3位加/减计数器，给出完整的时序图。说明计数器什么时候处于UP模式，什么时候处于DOWN模式。假设为上升沿触发。

0,1,2,3,2,1,2,3,4,5,6,5,4,3,2,1,0

13. 为具有如图8.74所示输入波形的74HC190加/减计数器，绘制出Q输出波形。一个二进制0位于数据输入上。开始于计数0000。

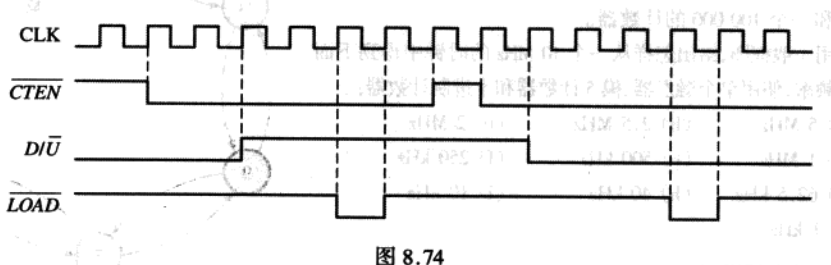


图 8.74

8.4节 同步计数器的设计

14. 确定图8.75中计数器的时序。

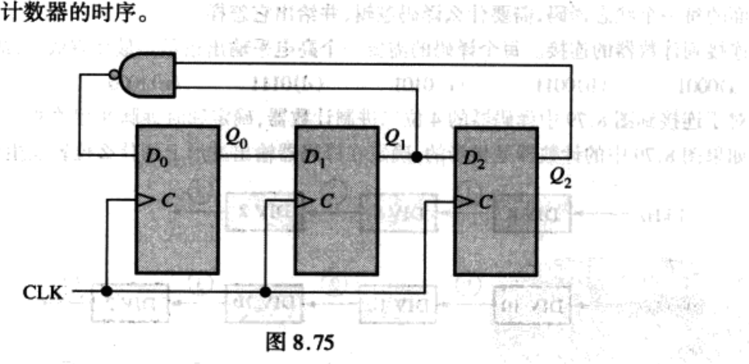


图 8.75

15. 确定图8.76中计数器的时序。计数器开始时为清零状态。
16. 设计一个计数器以产生下面的时序。使用J-K触发器。

00,10,01,11,00,…

17. 设计一个计数器以产生下面的时序。使用J-K触发器。

1,4,3,5,7,6,2,1,…

18. 设计一个计数器以产生下面的时序。使用J-K触发器。

0,9,1,8,2,7,3,6,4,5,0,…

19. 设计一个计数器,具有图 8.77 中状态图所给出的时序。

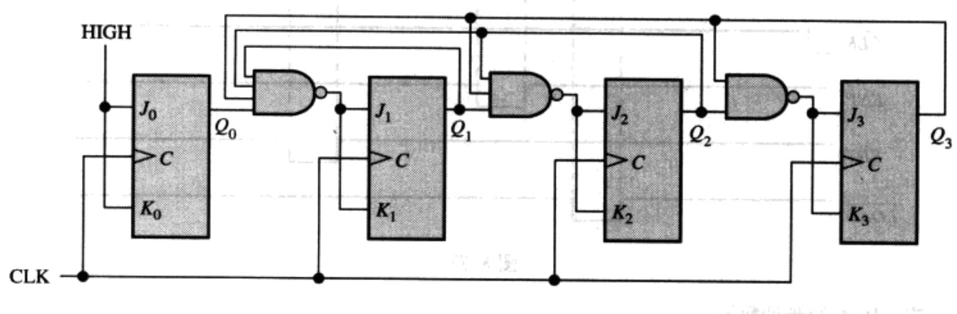


图 8.76

8.5 级联计数器

20. 对于图 8.78 中的级联计数器配置,确定由圆圈数字所表示的每个波形的频率,并且确定整体的模。

21. 扩展图 8.41 中的计数器以创建一个 10 000 分频的计数器和一个 100 000 的计数器。

22. 利用一般框图,给出怎样从一个 10 MHz 的时钟中得到下面的频率,使用单个触发器、模 5 计数器和十进制计数器:

(a) 5 MHz (b) 2.5 MHz (c) 2 MHz
(d) 1 MHz (e) 500 kHz (f) 250 kHz
(g) 62.5 kHz (h) 40 kHz (i) 10 kHz
(j) 1 kHz

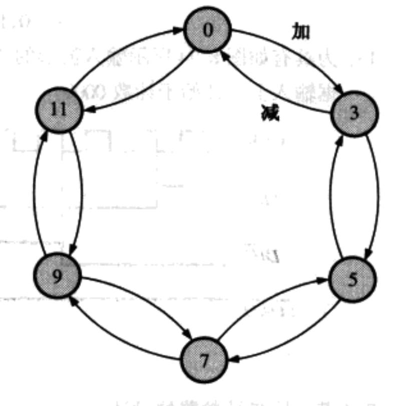


图 8.77

8.6 节 计数器译码

23. 一个给定的 BCD 十进制计数器只有 Q 输出可用,为给下面的每一个状态译码,需要什么译码逻辑,并给出它怎样连接到计数器的连接。每个译码的需要一个高电平输出指示。最高有效位(MSB)在左边。

(a)0001 (b)0011 (c)0101 (d)0111 (e)1000

24. 对于连接到图 8.79 中译码器的 4 位二进制计数器,确定和时钟脉冲相关的每个译码器的输出波形。

25. 如果图 8.79 中的计数器是异步的,确定在译码器输出波形上的什么位置发生译码假信号。

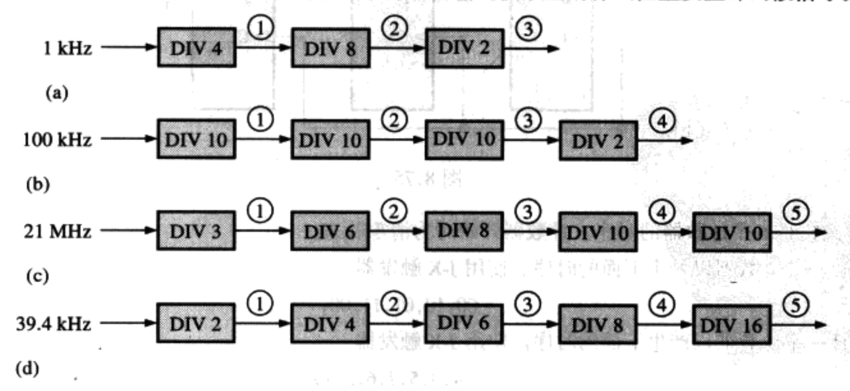


图 8.78

26. 修改图8.79中的电路，以消除译码假信号。

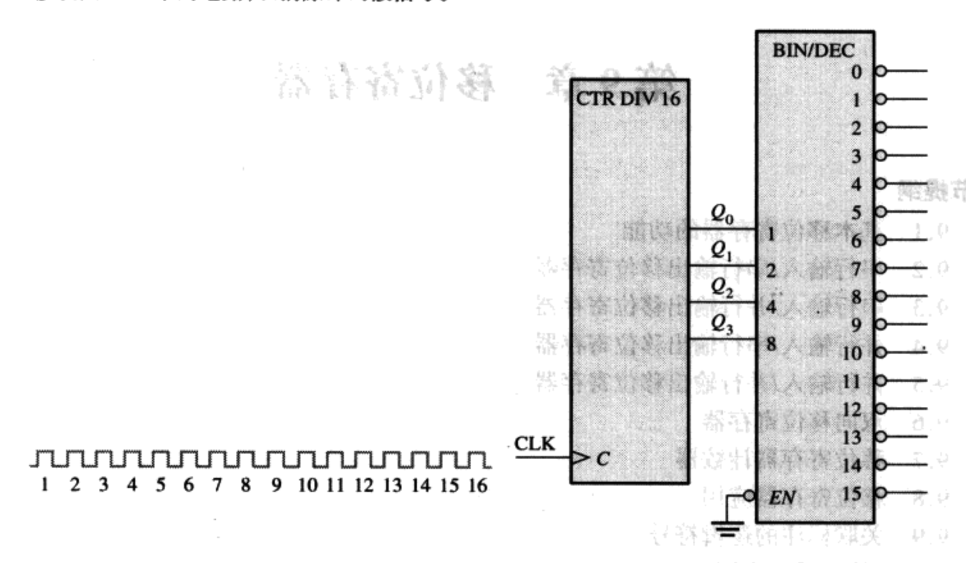


图8.79

27. 分析图8.45中的计数器，观察在译码门输出上是否有假信号。如果发生假信号，建议使用一种方式来消除它们。

28. 分析图8.46中的计数器，看看在译码门输出上是否有假信号。如果发生假信号，改变设计以消除它们。

8.7节 计数器应用

29. 假设图8.51中的数字时钟初始时被复位到了12点。确定在发生了62个60 Hz的脉冲之后，每个计数器的二进制状态。

30. 在图8.51中的数字时钟电路中，每个计数器的输出频率是多少？

31. 对于图8.54中的停车控制系统，在一个给定的24小时期间内，入口和出口传感器脉冲的图样如图8.79所示。如果在该时期之前车库中已经有53辆车，那么在24小时结束之后计数器的状态是什么？

自测题答案

1. (a) 2. (b) 3. (b) 4. (c) 5. (a) 6. (c) 7. (b) 8. (c) 9. (d) 10. (a) 11. (c) 12. (b) 13. (b) 14. (d)

第9章　移位寄存器

章节提纲

9.1　基本移位寄存器的功能

移位寄存器由若干触发器排列组成,在数字系统的应用中很重要,这些应用涉及数据的存储和移位。寄存器和计数器不同,除了某些很专业的应用之外,没有特定的状态时序。一般来说,寄存器仅仅用来存储外部数据源进来的数据(若干1和0)及对这些数据进行移位,它不具有象征意义上的特殊内部状态时序。

学完本节以后,应当能够

- 解释触发器怎样存储一个数据位
- 定义移位寄存器的存储容量
- 定义寄存器的移位能力

◇ *寄存器可以由一个或者多个用以存储和移位数据的触发器组成。*

寄存器是一个具有两种基本功能的数字电路,即数据存储和数据移动。寄存器的存储能力使得它成为一种重要的存储器设备。图9.1解释了在 D 触发器中存储1或者0这个存储概念。一个1加在数据输入上,同时加一个时钟脉冲使触发器置位,这样就存储了这个1。当输入上的1被移走之后,触发器还是保持在置位状态,因此存储了1。应用相似的过程,通过复位触发器来存储0,如图9.1所示。

寄存器的存储容量是它可以包含的数字数据的总位数(1和0)。移位寄存器中的每一级(触发器)都表示存储容量中的一个位,所以寄存器中的级数决定了它的存储容量。

寄存器的移位能力允许寄存器内数据从级到级的移动,或者根据所加的时钟脉冲,数据进入或者离开寄存器。图9.2解释了移位寄存器中的数据类型。方块图表示任意的4位寄存器,箭头指示数据移动的方向。

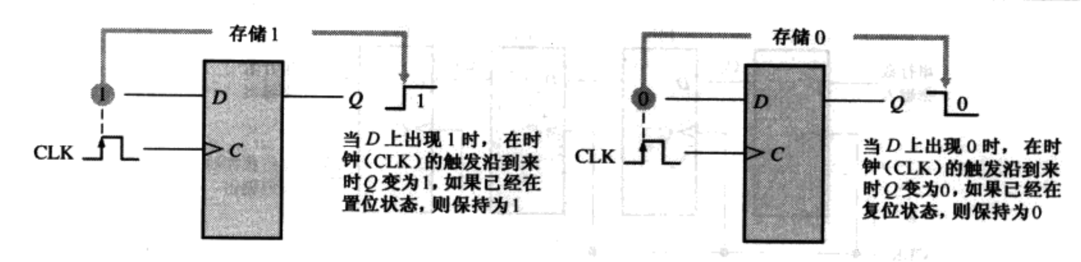


图9.1 触发器作为存储元件

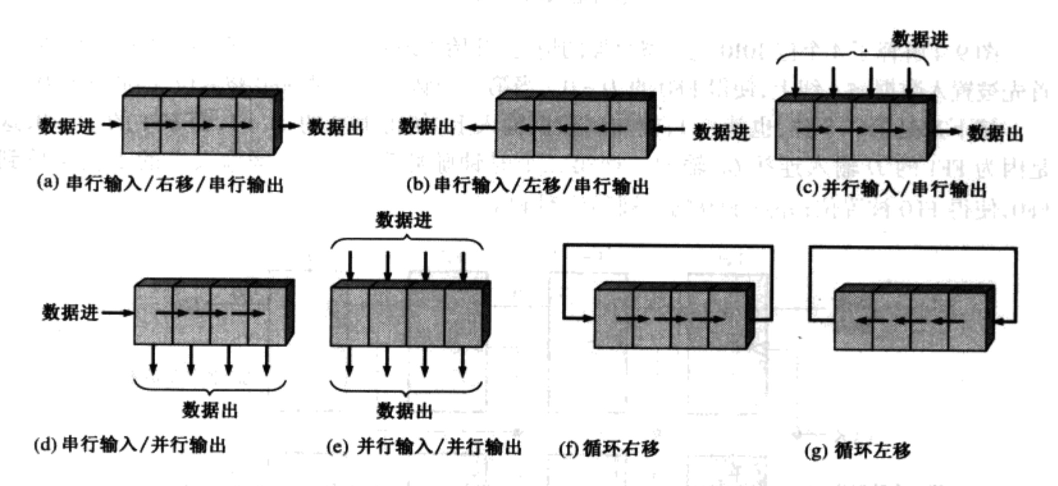


图9.2 移位寄存器的基本数据移动(用4个位进行解释，位按照箭头所指的方向移动)

9.2 串行输入/串行输出移位寄存器

串行输入/串行输出移位寄存器串行接收数据，也就是说，一条线上一次接收一个位。在输出上所产生的存储信息同样是串行形式的。

学完本节以后，应当能够

- 解释数据位怎样串行进入移位寄存器
- 描述数据位怎样在寄存器中移位
- 解释数据位怎样从移位寄存器中串行输出
- 设计并分析串行输入/串行输出寄存器的时序图

首先看看串行进入一个典型的移位寄存器的数据。图9.3给出了一个用 D 触发器实现的4位寄存器。其中有4个级，因此这个寄存器可以存储4位数据。



计算机小知识

经常需要对计算机中的内部寄存器清零。例如，可能先要在算术运算或者其他运算之前使寄存器清零。计算机中寄存器清零的一种方法是，使用软件减去寄存器自身所包含的内容。当然结果总是为0。例如，执行这个运算的计算机指令是 SUB AL,AL。使用这个指令，名称为 AL 的寄存器就被清零了。

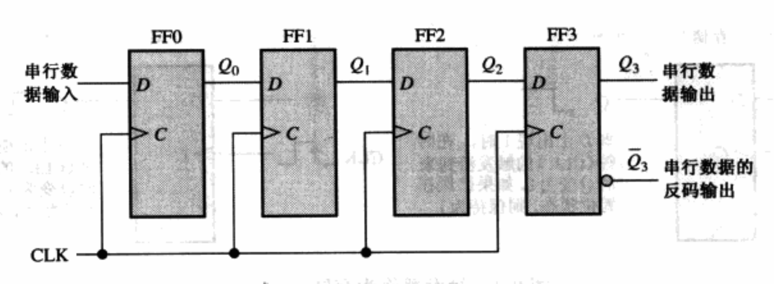


图 9.3　串行输入/串行输出移位寄存器

图 9.4 解释了 4 个位 1010 进入寄存器的情况,开始于最右边的位。寄存器初始时为清零。0 首先被置入数据输入线上,使得 FF0 的 $D=0$。当第一个脉冲来到时,FF0 被复位,因此存储 0。

接下来是第二个位,也就是 1,被加到数据输入上,使得 FF0 的 $D=1$,而 FF1 的 $D=0$,这是因为 FF1 的 D 输入连接 Q_0 输出。当第二个时钟脉冲到来时,数据输入上的 1 被移位到 FF0,使得 FF0 被置位;并且 FF0 的 0 被移位到 FF1。

图 9.4　4 个位(1010)串行进入寄存器

第三个位，也就是 0，现在被置入数据输入线上，同时加上了一个时钟脉冲。这个 0 进入 FF0，FF0 中存储的 1 移位到 FF1 中，而 FF1 所存储的 0 移位到 FF2 中。

最后一位，也就是 1，现在被加到数据输入上，同时增加了一个时钟脉冲。这次 1 进 FF0，而存储在 FF0 中的 0 移位到 FF1 中，存储在 FF1 中的 1 移位到 FF2，存储在 FF2 中的 0 移位到 FF3。这就完成了 4 位串行进入移位寄存器的过程，在寄存器中它们可以被存储无限长的时间，只要触发器有直流电源连接。

◇ 对于串行数据来说，一次传递一个位。

如果想要得到寄存器输出的数据，数位就必须串行移出并在 Q_3 输出取走，如图 9.5 所示。在刚才描述的数据输入运算中的 CLK4 之后，最右边的位 0 就会出现在 Q_3 输出。当时钟脉冲 CLK5 到来时，第二个位出现在 Q_3 输出。时钟脉冲 CLK6 把第三个位移到输出，而 CLK7 把第四个位移位到输出。当原始的 4 个位移出寄存器时，更多的位可以移入。如图所示，全部移入 0。

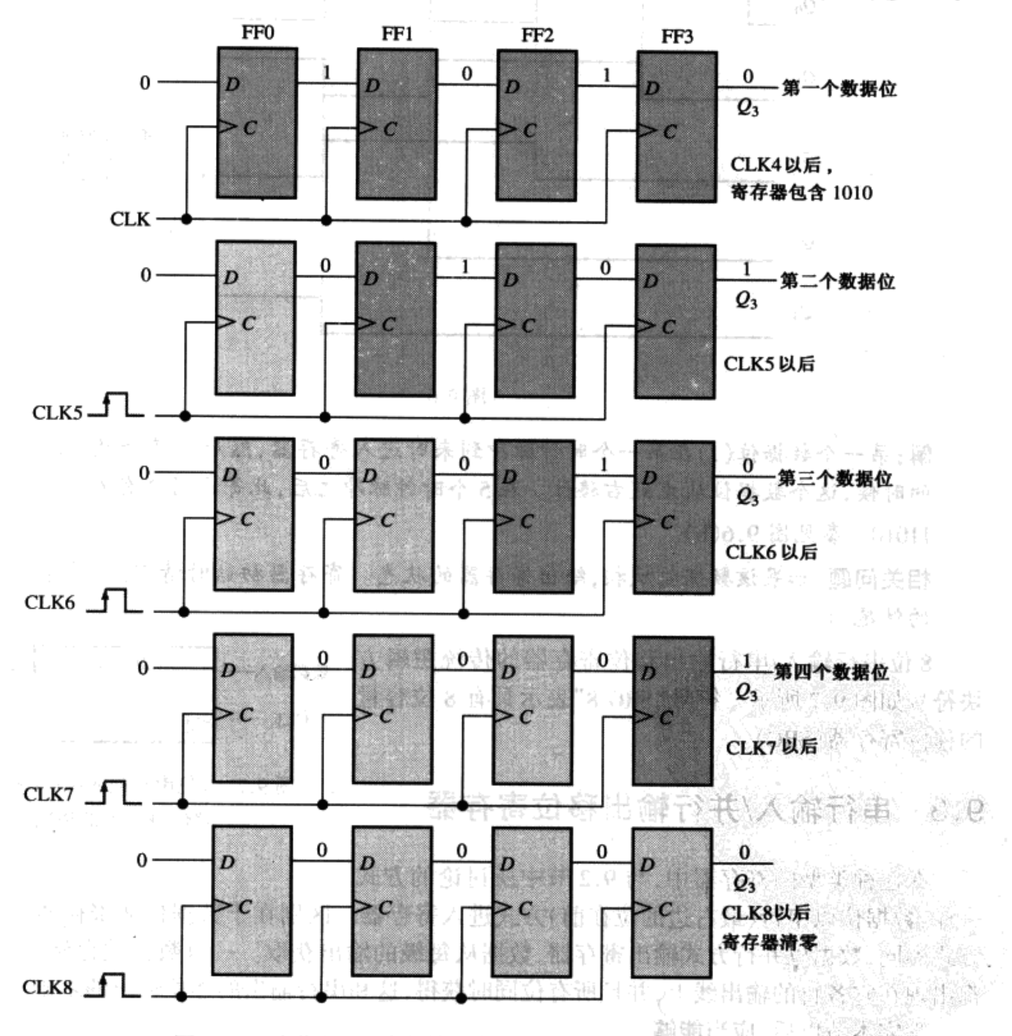


图 9.5　4 个位(1010)串行输出寄存器，然后全部用零替代

例 9.1 对于指定的数据输入和时钟波形,给出图 9.6(a)中的 5 位寄存器的状态。假设该寄存器初始时被清零(全 0)。

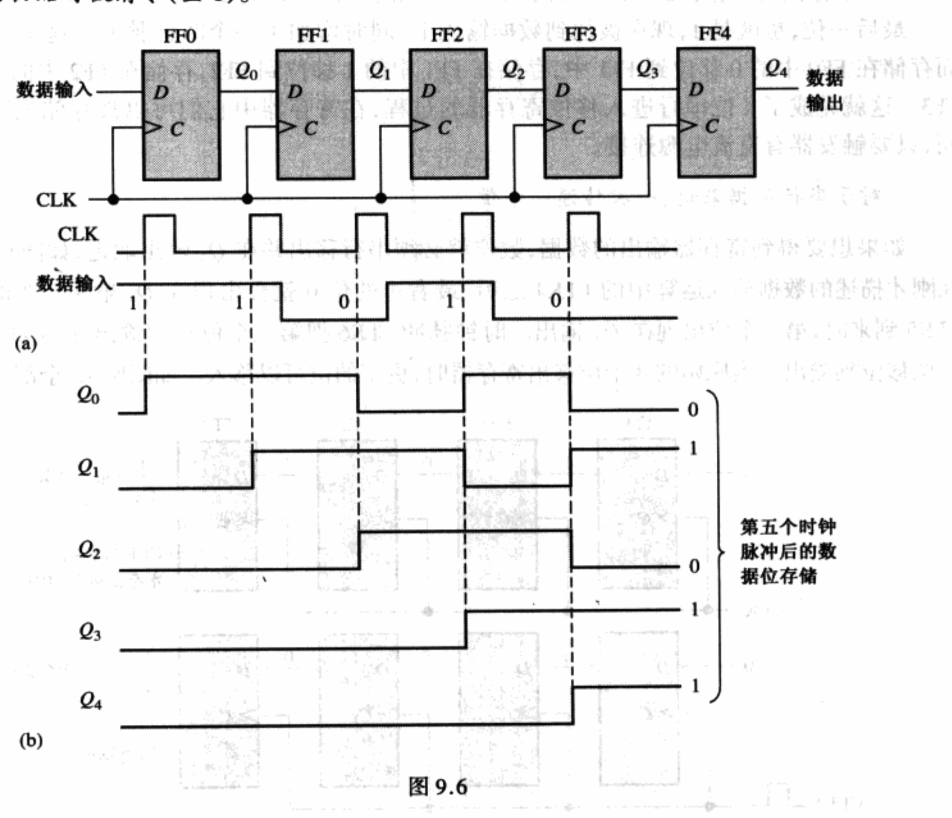


图 9.6

解: 第一个数据位(1)在第一个时钟脉冲到来时进入寄存器,然后在其余数位进入和移位的时候,这个数据位从左到右移位。在 5 个时钟脉冲之后,此寄存器含有 $Q_4Q_3Q_2Q_1Q_0$ = 11010。参见图 9.6(b)。

相关问题: 如果该数据位反相,给出寄存器的状态。寄存器初始时清零。(答案参见本章的结尾。)

8 位串行输入/串行输出移位寄存器的传统逻辑方块符号如图 9.7 所示。符号"SRG 8"表示具有 8 位容量的移位寄存器(SRG)。

数据输入 SRG 8 Q_7
CLK C $\bar{Q}_7$

图 9.7 8 位串行输入/串行输出移位寄存器的逻辑符号

9.3 串行输入/并行输出移位寄存器

在这种类型的寄存器中,与 9.2 节中所讨论的方式一样,数据位以串行(最右边的位在前)方式进入寄存器。区别在于数据位从寄存器中取出的方式不同;数据以并行方式输出寄存器,数据从每级的输出获取。一旦数据被存储后,每个位都出现在它各自的输出线上,并且所有位同时获得,这和串行输出的逐位输出的基础不同。

学完本节以后,应当能够

- 解释数据位怎样并行从移位寄存器中输出
- 比较串行输出和并行输出
- 讨论74HC 1648位移位寄存器
- 设计并分析串行输入/并行输出寄存器的时序图

图9.8给出了一个4位串行输入/并行输出移位寄存器及它的逻辑方块符号。

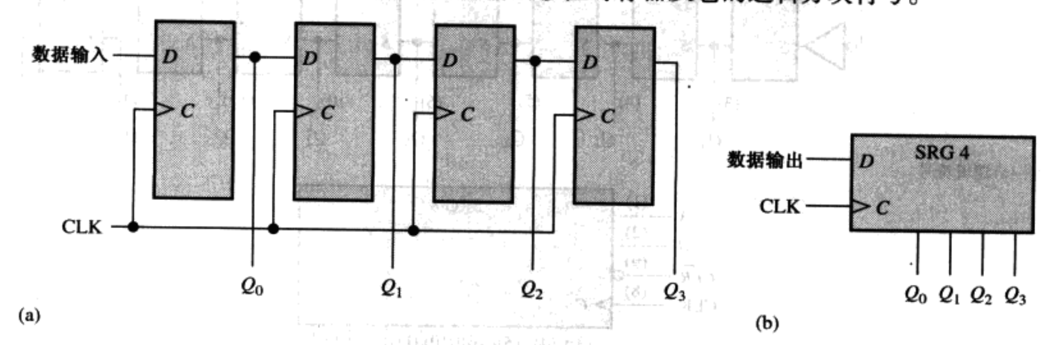


图9.8 串行输入/并行输出移位寄存器

例9.2 对于图9.9(a)中的数据输入和时钟波形,给出此4位寄存器(SRG4)的状态。寄存器初始时全为1。

解:该寄存器在4个时钟脉冲以后,状态为0110。参见图9.9(b)。

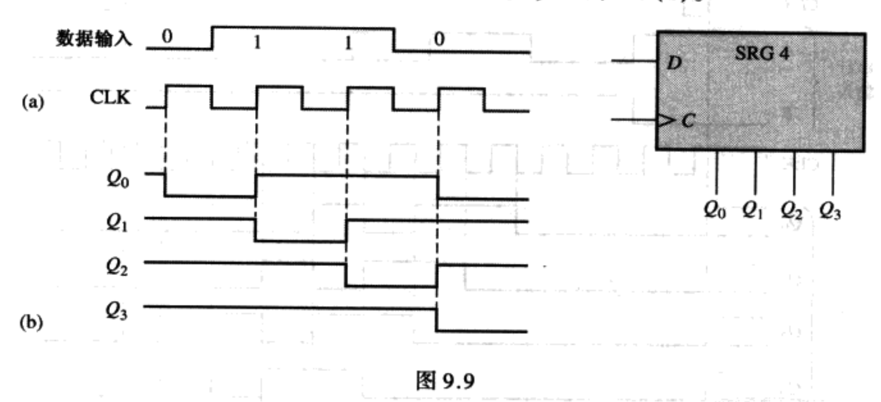


图9.9

相关问题:如果数据输入在第四个时钟脉冲以后仍然保持0,那么在三个附加的时钟脉冲以后,寄存器的状态是什么?

74HC164 8位串行输入/并行输出移位寄存器

74HC164是具有串行输入/串行输出运算的IC移位寄存器的一个例子。逻辑图如图9.10(a)所示,典型的逻辑方块符号如图9.10(b)所示。注意这个芯片具有两个门控串行输入A和B及一个低电平有效的清零($\overline{CLR}$)输入。并行输出为Q_0到Q_7。

74HC164的一个样例时序图如图9.11所示。注意输入A上的串行输入数据在输入B变为高电平以后,移位并通过寄存器。

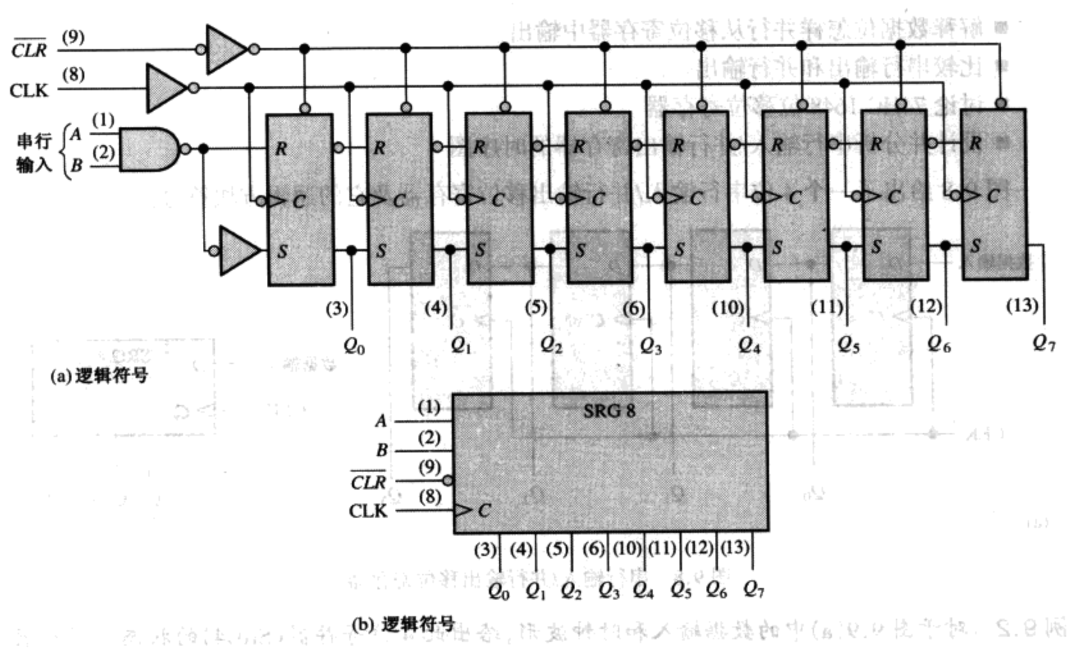


图 9.10　74HC164 8 位串行输入/并行输出移位寄存器

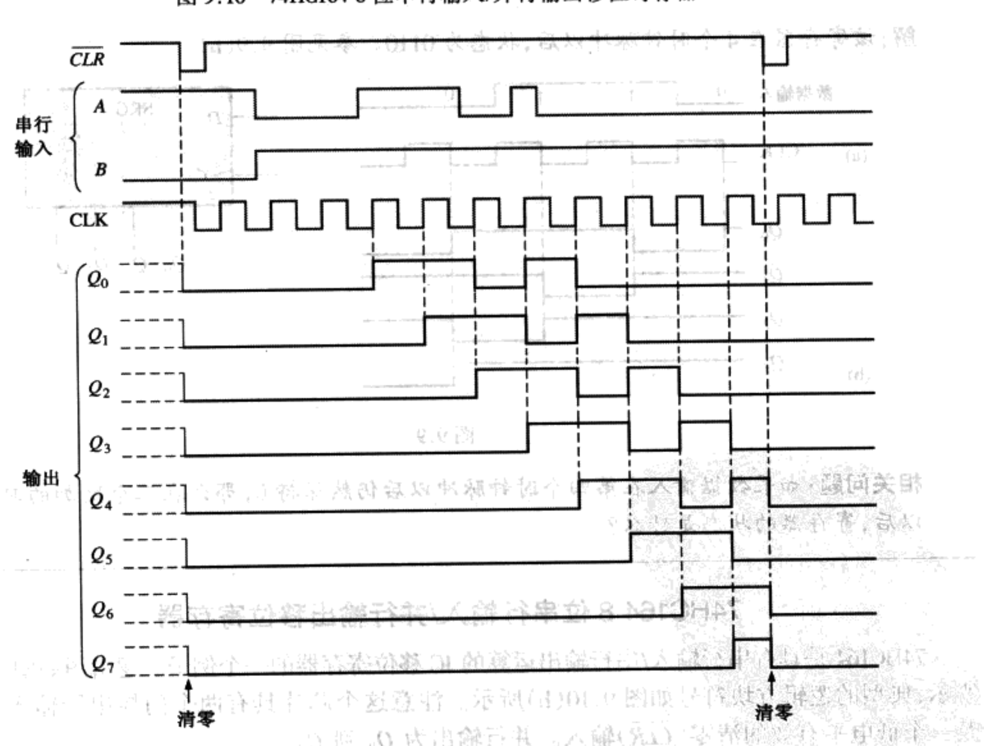


图 9.11　74HC164 移位寄存器的样例时序图

9.4 并行输入/串行输出移位寄存器

对于具有并行数据输入的寄存器来说,数位同时进入位于并行线上相应的级,而不是像串行数据输入那样在一条线上基于逐位的方法。一旦数据完全存储在寄存器中,串行输出和9.2节所讨论的相同。

在学完本节以后,应当能够

- 解释数据数位怎样并行进入移位寄存器
- 比较串行输入和并行输入
- 讨论74HC165 8位并行置数移位寄存器
- 设计并分析并行输入/串行输出寄存器的时序图

◇ 对于并行数据来说,一次传送多位数据。

图9.12解释了一个4位并行输入/串行输出移位寄存器和一个典型的逻辑符号。注意有4条数据输入线 D_0、D_1、D_2 和 D_3,以及一个 $SHIFT/\overline{LOAD}$(移位/置数)输入,其允许4位数据并行进入寄存器中。当 $SHIFT/\overline{LOAD}$ 为低电平时,门 G_1 到 G_4 开启,允许每个数据位分别加到相应触发器的 D 输入端。当时钟脉冲到来时,$D_1=1$ 的触发器将置位,而 D_0 的触发器将复位,因此就同时存储了所有的4个位。

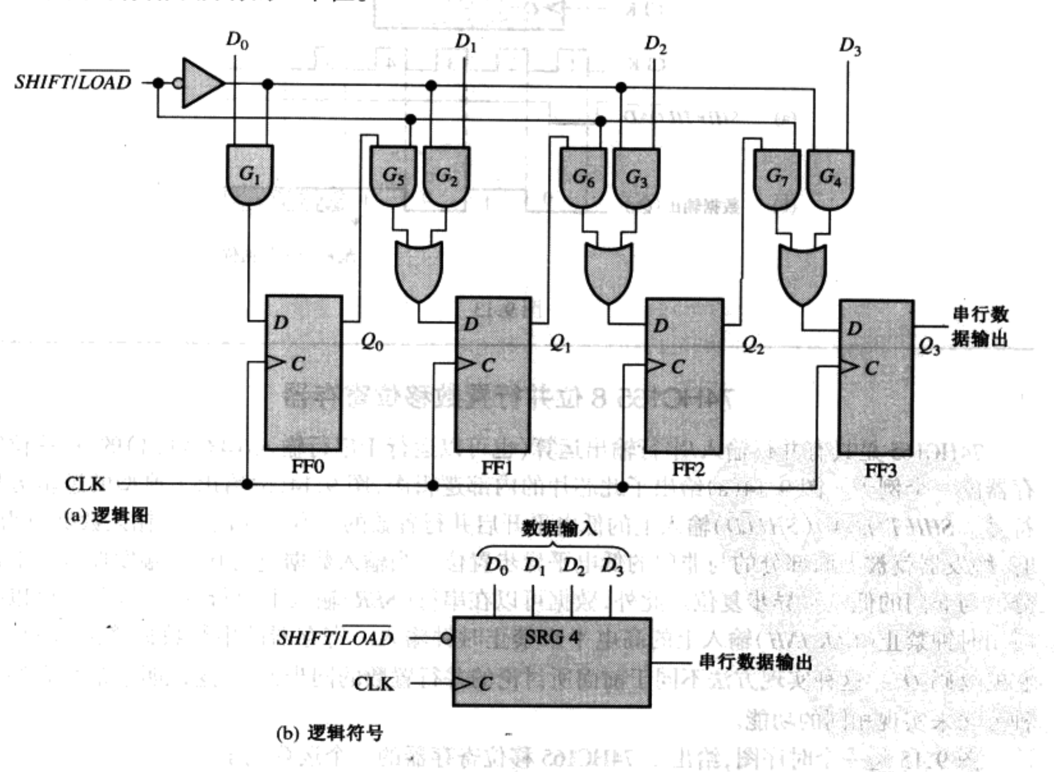


图 9.12 4位并行输入/串行输出移位寄存器

当 $SHIFT/\overline{LOAD}$为高电平时,G_1 到 G_4 被禁止,而门 G_5 到 G_7 被开启,这就允许数据位从一个级向右移位到下一级。或门允许正常的移位或者并行数据进入的运算,这取决于 $SHIFT/\overline{LOAD}$输入上的电平所开启的与门。注意,FF0 有一个与门禁止并行输入,即 D_0。这里不需要与/或连接,因为没有串行数据输入。

例 9.3 对于图 9.13(a)所示的并行输入数据、时钟及 $SHIFT/\overline{LOAD}$波形,给出 4 位寄存器的数据输出波形。参见图 9.12(a)的逻辑图。

解:时钟脉冲到来时,并行数据($D_0D_1D_2D_3=1010$)被置入寄存器中,使得 Q_3 为 0。时钟脉冲 2 到来时,来自 Q_2 的 1 被移位到 Q_3;时钟脉冲 3 到来时,0 被移位到 Q_3;在时钟脉冲 4,最后一个数据位(1)被移位到 Q_3;在时钟脉冲 5 上,所有的数据位已经被移出,只有 1 保存在寄存器中(假设 D 输入保持 1)。参见图 9.13(b)。

相关问题:如果并行数据为 $D_0D_1D_2D_3=0101$,对于如图 9.13(a)所示的时钟和 $SHIFT/\overline{LOAD}$输入,给出数据输出波形。

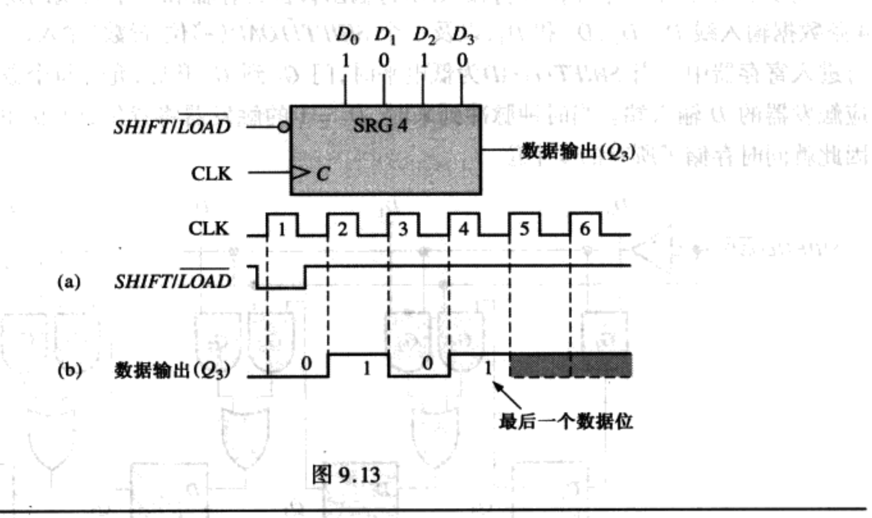
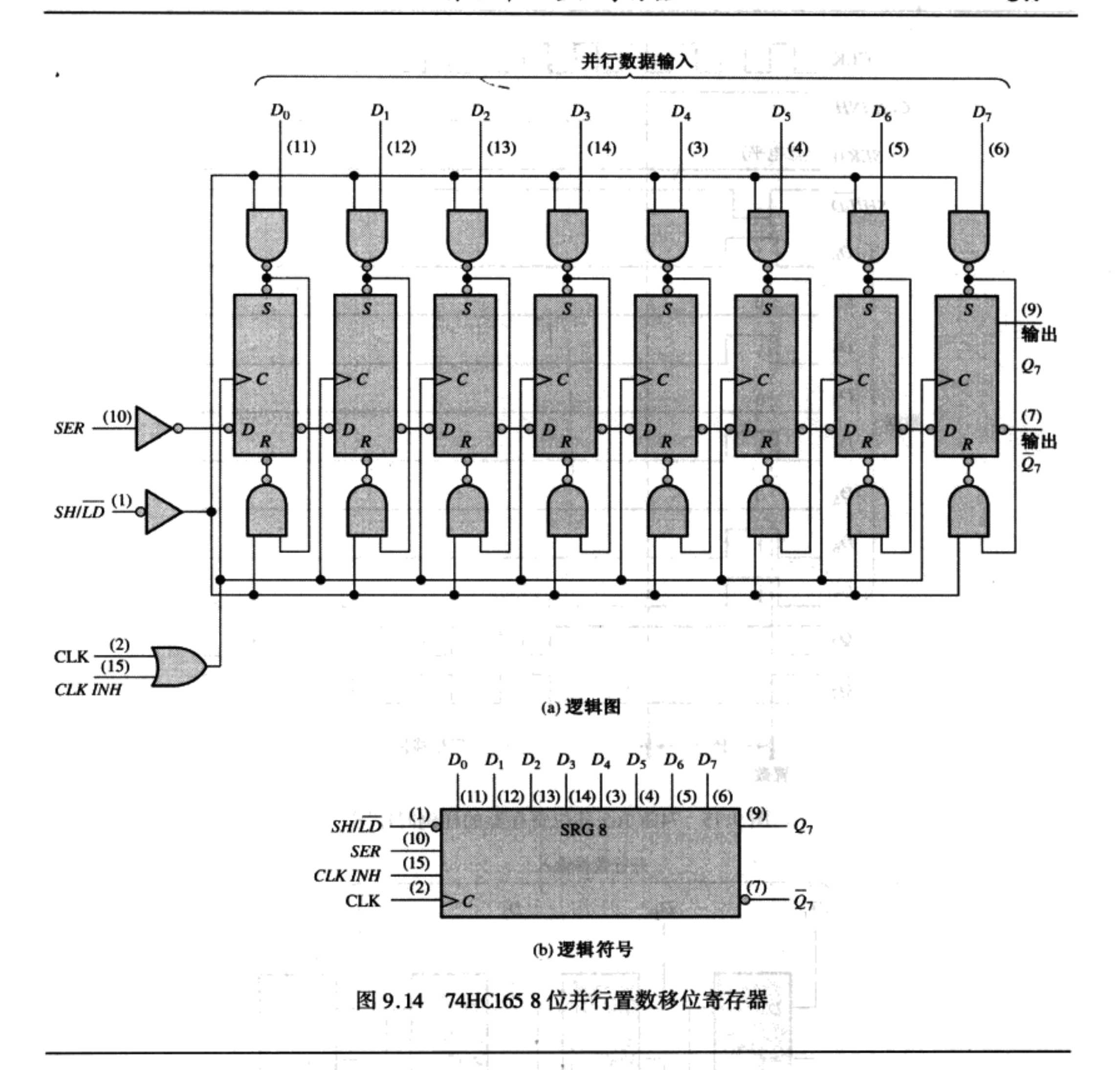


图 9.13

74HC165 8 位并行置数移位寄存器

74HC165 是具有并行输入/串行输出运算(也可以运行于串行输入/串行输出)的 IC 移位寄存器的一个例子。图 9.14(a)给出了此芯片的内部逻辑图,图 9.14(b)给出了典型的逻辑方块符号。$SHIFT/\overline{LOAD}$($SH/\overline{LD}$)输入上的低电平开启并行置数的所有与非门。当输入数据位为 1 时,触发器就被上面部分的与非门的低电平异步置位。当输入数据位为 0 时,触发器被底下部分的与非门的低电平异步复位。此外,数据可以在串行(*SER*)输入上串行进入。同样,可以随时用时钟禁止(*CLK INH*)输入上的高电平来禁止时钟输入。寄存器的串行数据输出为 Q_7 和它的反码 $\overline{Q}_7$。这种实现方法不同于前面所讨论的并行置数的同步方法,这说明通常会有好几种方法来实现相同的功能。

图 9.15 是一个时序图,给出了 74HC165 移位寄存器的一个运算例子。



(a) 逻辑图

(b) 逻辑符号

图 9.14 74HC165 8 位并行置数移位寄存器

9.5 并行输入/并行输出移位寄存器

并行数据输入在 9.4 节已经描述了,并且数据的并行输出也在前面讨论过了。并行输入/并行输出可以使用这两种方法。在全部数据位同时进入之后,这些数据位就会即刻出现在并行输出端。

学完本节以后,应当能够

- 讨论 74HC195 4 位并行存取移位寄存器
- 设计并分析并行输入/并行输出寄存器的时序图

图 9.16 给出了一个并行输入/并行输出寄存器。

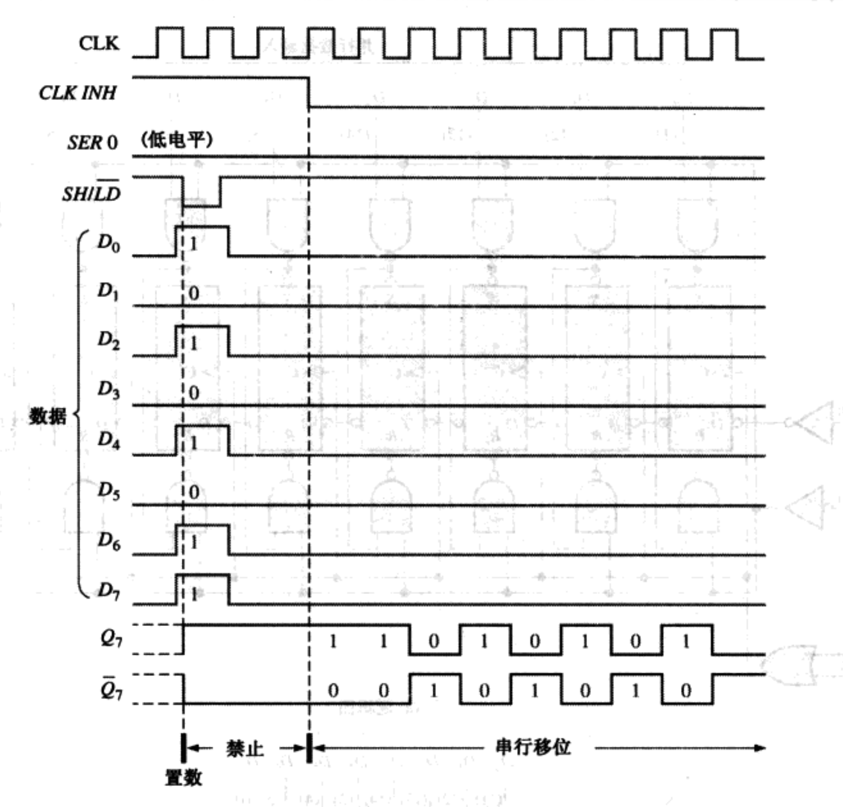


图 9.15　74HC165 移位寄存器的样例时序图

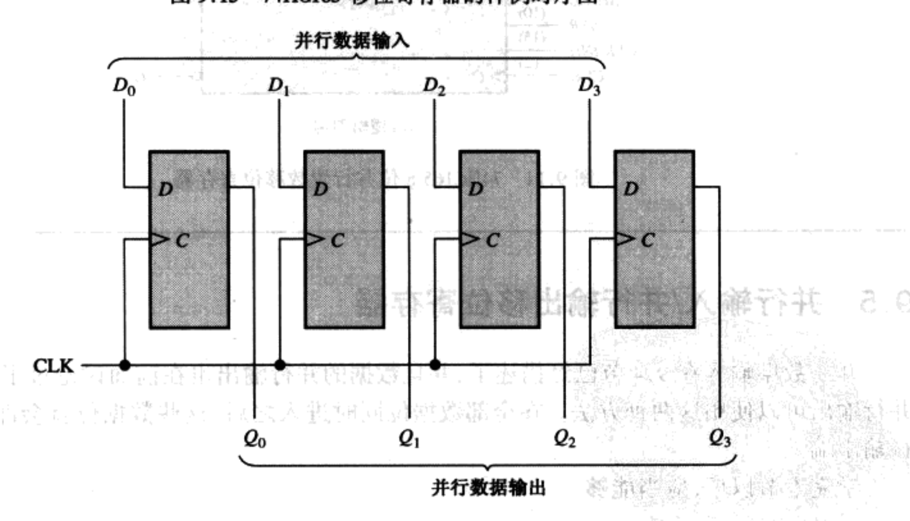


图 9.16　并行输入/并行输出寄存器

74HC195 4 位并行存取移位寄存器

74HC195 可以用于并行输入/并行输出运算。因为它也可以具有串行输入，所以也可以用

做串行输入/串行输出及串行输入/并行输出运算。通过把 Q_3 用做输出，它可以用于并行输入/串行输出运算。典型的逻辑方块符号如图 9.17 所示。

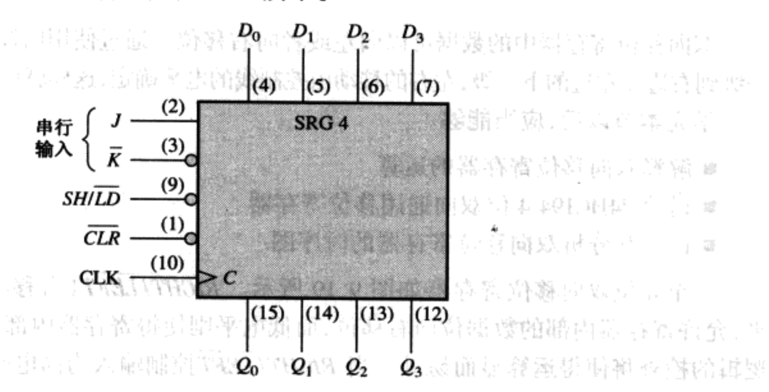


图 9.17　74HC195 4 位并行存取移位寄存器

当 $SHIFT/\overline{LOAD}$ 输入（$SH/\overline{LD}$）为低电平时，并行输入上的数据在时钟的上升沿转换到来时同步进入。当 $SH/\overline{LD}$ 为高电平时，存储的数据将会随着时钟向右同步移位（Q_0 到 Q_3）。输入 J 和 $\overline{K}$ 是寄存器第一级（Q_0）的串行数据输入；Q_3 可以用做串行输出数据。低电平有效清零输入是同步的。

图 9.18 中的时序图解释了此寄存器的运算。

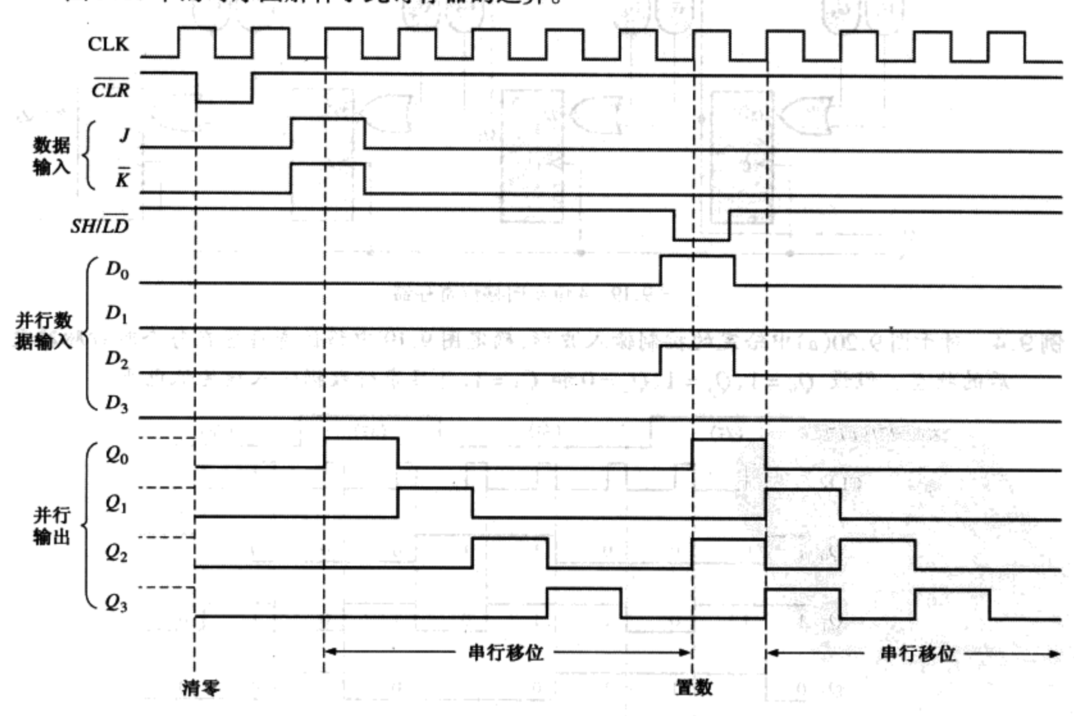


图 9.18　74HC195 移位寄存器的简单时序图

9.6 双向移位寄存器

双向移位寄存器中的数据可以向左或者向右移位。通过使用门控逻辑使得数据位的传送从一级到右边或左边的下一级,左右的移动由控制线的电平确定,这样就可以实现数据的双向移位。

学完本节以后,应当能够

- 解释双向移位寄存器的运算
- 讨论 74HC194 4 位双向通用移位寄存器
- 设计并分析双向移位寄存器的时序图

一个 4 位双向移位寄存器如图 9.19 所示。$RIGHT/\overline{LEFT}$(右移/左移)控制输入上的高电平,允许寄存器内部的数据位向右移位,而低电平则使得寄存器内部的数据位向左移位。门控逻辑的检查将使得运算显而易见。当 $RIGHT/\overline{LEFT}$控制输入为高电平时,门 G_1 到 G_4 开启,而每个触发器 Q 输出的状态传送到下一个触发器的 D 输入上。当时钟脉冲到来时,数据位向右移动一个位置。当 $RIGHT/\overline{LEFT}$控制输入为低电平时,门 G_5 到 G_8 开启,而每个触发器 Q 输出传送到前一个触发器的 D 输入上。当时钟脉冲到来时,数据位就会向左移动一个位置。

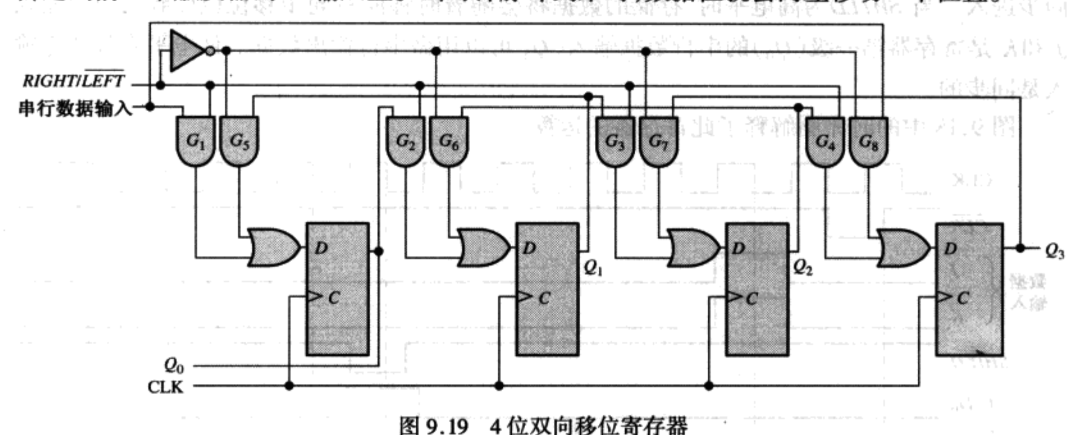


图 9.19 4 位双向移位寄存器

例 9.4 对于图 9.20(a)中给定的控制输入波形,确定图 9.19 中移位寄存器在每个时钟脉冲之后的状态。假设 $Q_0=1$、$Q_1=1$、$Q_2=0$ 和 $Q_3=1$,并且串行数据输入线为低电平。

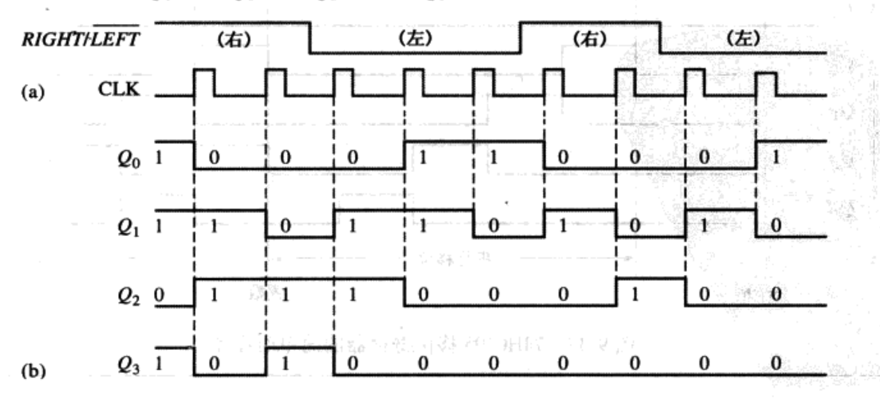


图 9.20

解：参见图 9.20(b)。

相关问题：把 *RIGHT*/$\overline{LEFT}$ 波形反相，确定图 9.19 中移位寄存器在每个时钟脉冲之后的状态。

74HC194 4 位双向通用移位寄存器

74HC194 是集成电路形式中通用双向移位寄存器的一个例子。通用移位寄存器同时具有串行和并行输入和输出的能力。逻辑方块符号如图 9.21 所示，而样例时序图如图 9.22 所示。

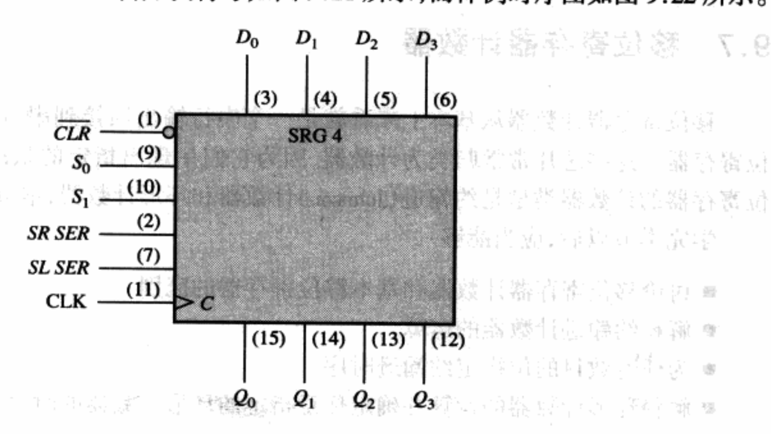


图 9.21　74HC194 4 位双向通用移位寄存器

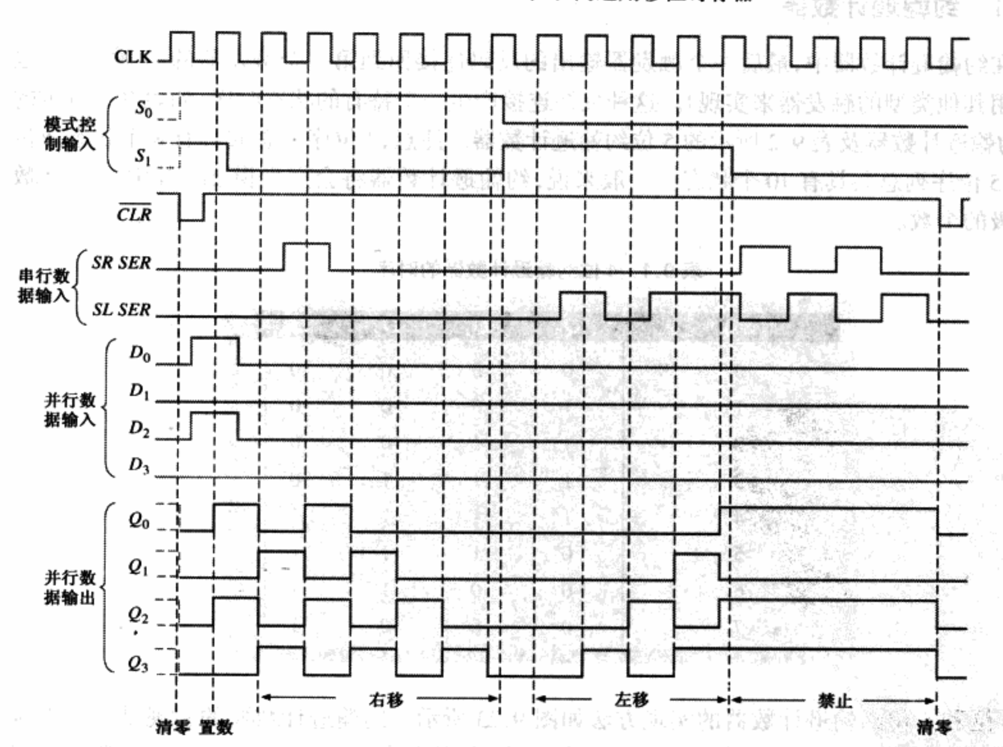


图 9.22　74HC194 移位寄存器的样例时序图

同步于时钟上升沿转换的并行置数,是通过第4个数据位加到并行输入及把一个高电平加到 S_0 和 S_1 输入上来实现的。当 S_0 为高电平和 S_1 为低电平时,向右移位和时钟的上升沿同步完成。模式中的串行数据在向右移位串行输入(*SR SER*)上进入。当 S_0 为低电平和 S_1 为高电平时,数据位左移同步于时钟,而新数据在左移串行输入(*SL SER*)上进入。输入 *SR SER* 进入 Q_0 级,而 *SL SER* 进入 Q_3 级。

9.7 移位寄存器计数器

移位寄存器计数器从基本上来看就是一个串行输出回接到串行输入,产生特殊时序的移位寄存器。这些芯片常常归类为计数器,因为它们呈现出指定的状态时序。最常见的两种移位寄存器的计数器类型是约翰逊(Johnson)计数器和环形计数器,本节将会介绍它们。

学完本节以后,应当能够

- 讨论移位寄存器计数器和基本移位寄存器的区别
- 解释约翰逊计数器的运算
- 为任意数目的位指定约翰逊时序
- 解释环形计数器的运算并确定任意指定的环形计数器的时序

9.7.1 约翰逊计数器

在约翰逊计数器中,最后一个触发器输出的反码连接回到第一个触发器的 *D* 输入上(也可以用其他类型的触发器来实现)。这种反馈连接产生一个特有的状态时序,如表9.1所示的4位约翰逊计数器及表9.2所示的5位约翰逊计数器。注意,4位序列总共具有8个状态或位模式,5位序列总共具有10个状态。一般来说,约翰逊计数器将会产生模 $2n$,其中 n 是计数器中级的个数。

表 9.1 4位约翰逊计数器的时序

时钟脉冲	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

4位和5位约翰逊计数器的实现方法如图9.23所示。约翰逊计数器的实现是非常直接的,除了级数不同之外都是一样的。每一级的 *Q* 输出都连接到下一级的 *D* 输入上(假设使用

D 触发器)。唯一的例外是最后一级的 $\overline{Q}$ 输出连接回到第一级的 D 输入上。如表 9.1 和表 9.2所给出的时序那样,计数器将从左到右“填充”1,然后再“填充”0。

4 位和 5 位约翰逊计数器的时序运算图分别如图 9.24 和图 9.25 所示。

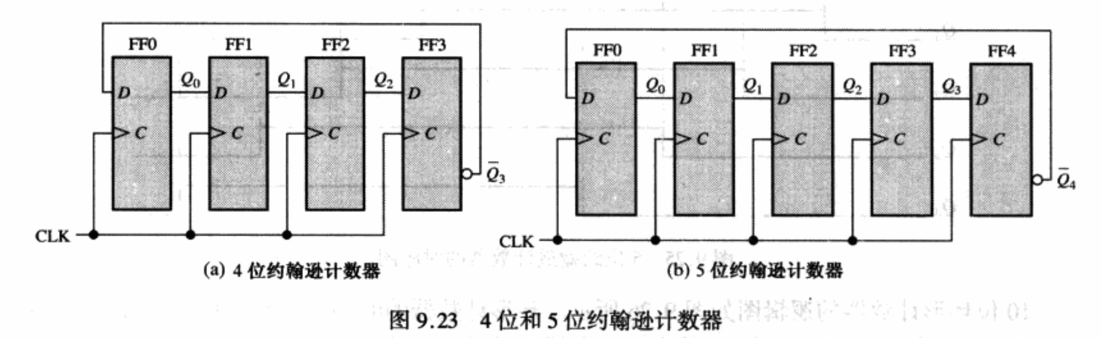


图 9.23　4 位和 5 位约翰逊计数器

表 9.2　5 位约翰逊计数器的时序

时钟脉冲	Q_0	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1

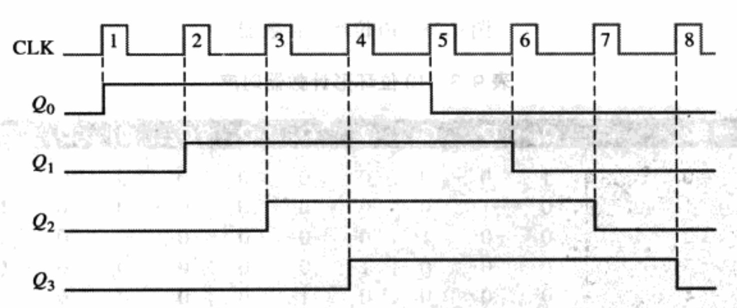


图 9.24　4 位约翰逊计数器的时序图

9.7.2　环形计数器

环形计数器(ring counter)为时序中的每个状态都使用一个触发器。它具有的优点是不需要译码门。在 10 位环形计数器的情况下,每个十进制数字都有唯一的一个输出。

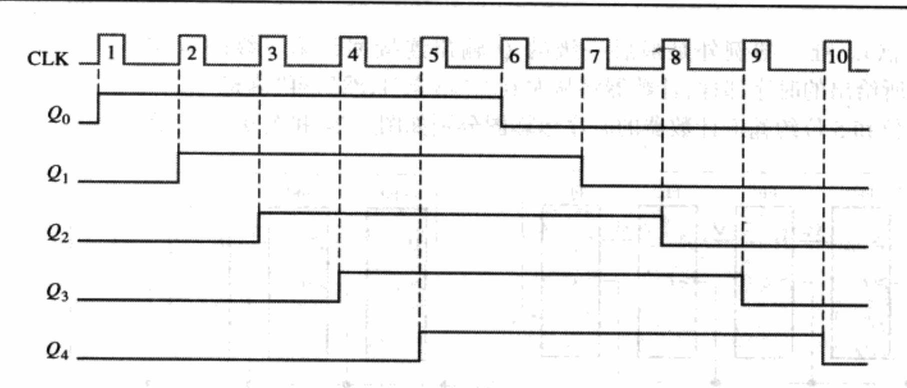


图 9.25　5 位约翰逊计数器的时序图

10 位环形计数器的逻辑图如图 9.26 所示。环形计数器的时序如表 9.3 所示。初始时,一个 1 被预置到第一个触发器中,而其余的触发器都被清零。注意级间连接和约翰逊计数器是一样的,不同的是 Q 而不是 $\overline{Q}$ 从最后一级反馈回去。计数器的 10 个输出直接指明了时钟脉冲的十进制计数。例如,Q_0 上的 1 表示 0,Q_1 上的 1 表示 1,Q_2 上的 1 表示 2,而 Q_3 上的 1 表示 3,以此类推。应当验证 1 总是保持在计数器中,并且简单地“围绕圆环”移位,每个时钟脉冲到来时移动一级。

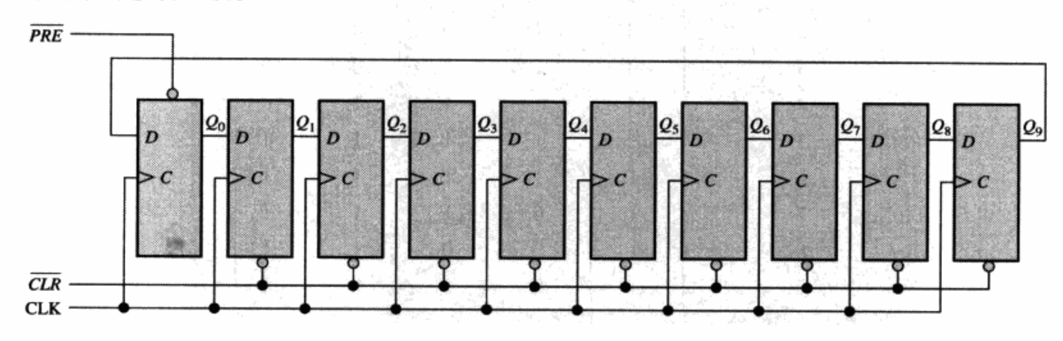


图 9.26　10 位环形计数器

表 9.3　10 位环形计数器时序

时钟脉冲	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1

使计数器中具有多于一个的单个 1,就可以得到修改的时序,如例 9.5 所示。

例 9.5 如果和图 9.26 相似的一个 10 位环形计数器的初始状态为 1010000000，确定每个 Q 输出的波形。

解：参见图 9.27。

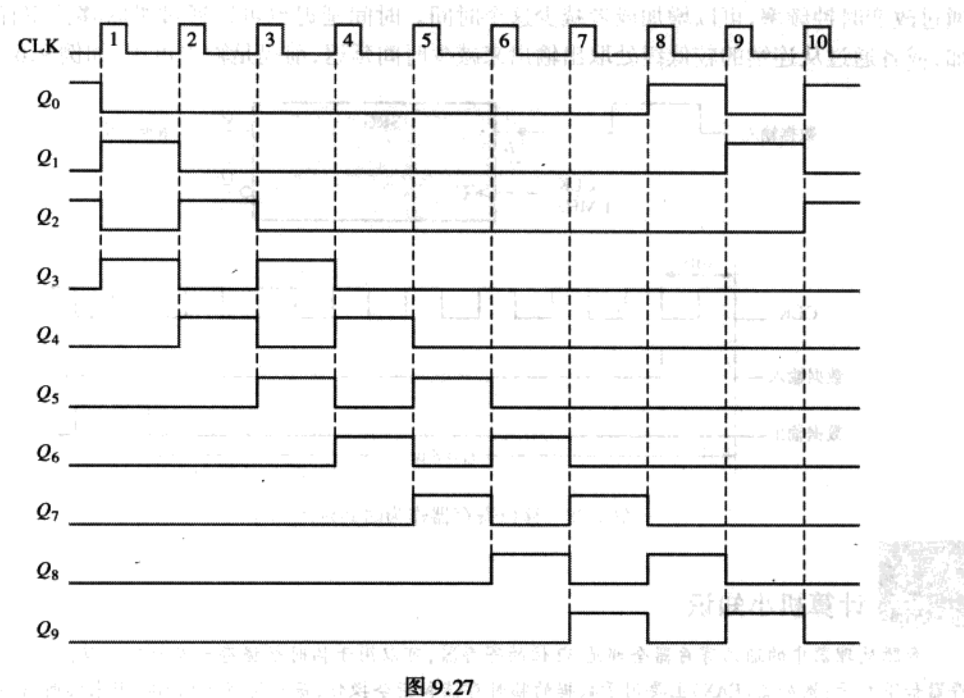


图 9.27

相关问题：如果一个 10 位环形计数器具有初始状态 0101001111，确定每个 Q 输出的波形。

9.8 移位寄存器应用

在许多类型的应用中都会发现移位寄存器，本节将介绍其中的一小部分。

学完本节以后，应当能够

- 使用移位寄存器产生时间延迟
- 使用 74HC195 移位寄存器来实现指定的环形计数器时序
- 讨论移位寄存器怎样用以数据的串行到并行的转换
- 定义 *UART*
- 解释键盘译码器的运算及寄存器怎样在这个应用中使用

9.8.1 时间延迟

串行输入/串行输出移位寄存器可以用来提供从输入到输出的时间延迟，此时间延迟是寄存器中级数(n)和时钟频率的函数。

当一个数据脉冲加在串行输入时，如图 9.28 所示(A 和 B 连接在一起)，它就在时钟脉冲

的触发边沿进入第一级。然后在每一个相继的时钟脉冲作用下,从一级移位到下一级,直至在 n 个时钟周期之后出现在串行输出上。这个时间延迟运算如图 9.28 所示,其中使用了一个具有 1 MHz 时钟频率的 8 位串行输入/串行输出移位寄存器,实现 8 μs 的时间延迟 $t_d(8\times 1\ \mu s)$。通过改变时钟频率,可以增加或者减少这个时间。时间延迟也可以通过级联移位寄存器来增加,或者通过从连续的较低级处取出输出来减少时间延迟,前提是输出可用,如例 9.6所示。

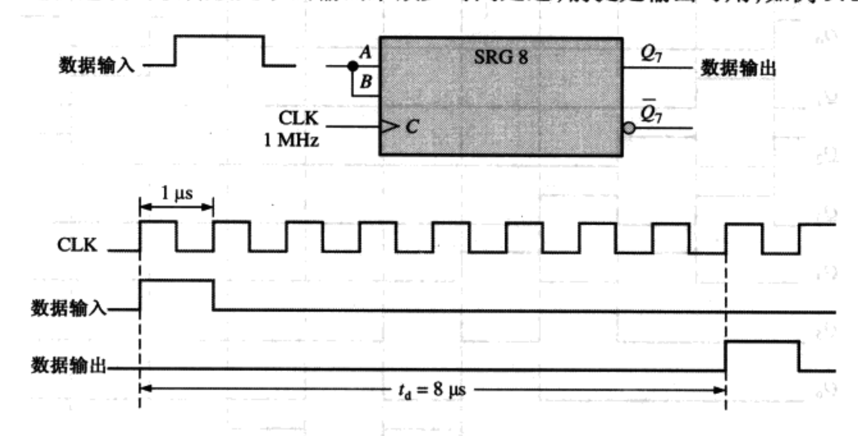


图 9.28 移位寄存器作为时间延迟芯片



计算机小知识

奔腾处理器中的通用寄存器全部是 32 位的寄存器,可以用于临时数据存储及特殊用途。其中的四种寄存器如下所示:累加器(EAX)主要用于数据的临时存储和指令操作;基址寄存器(EBX)用来临时存储数据;计数寄存器(ECX)主要用来确定某个循环、字符串、移位或者循环操作的重复次数;数据寄存器(EDX)一般用来临时存储数据。

例 9.6 确定图 9.29 中串行输入和每个输出之间的时间延迟的大小。绘制一个时序图来说明。

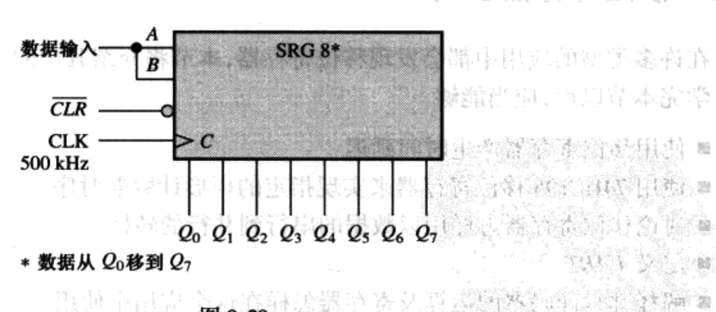


图 9.29

解: 时钟周期是 2 μs。因此,时间延迟能够以 2 μs 的增量在最小值 2 μs 到最大值 16 μs 之间增加或者减少,如图 9.30 所示。

相关问题: 要得到图 9.29 中输入到 Q_7 输出之间的 24 μs 时间延迟,确定所需要的时钟频率。

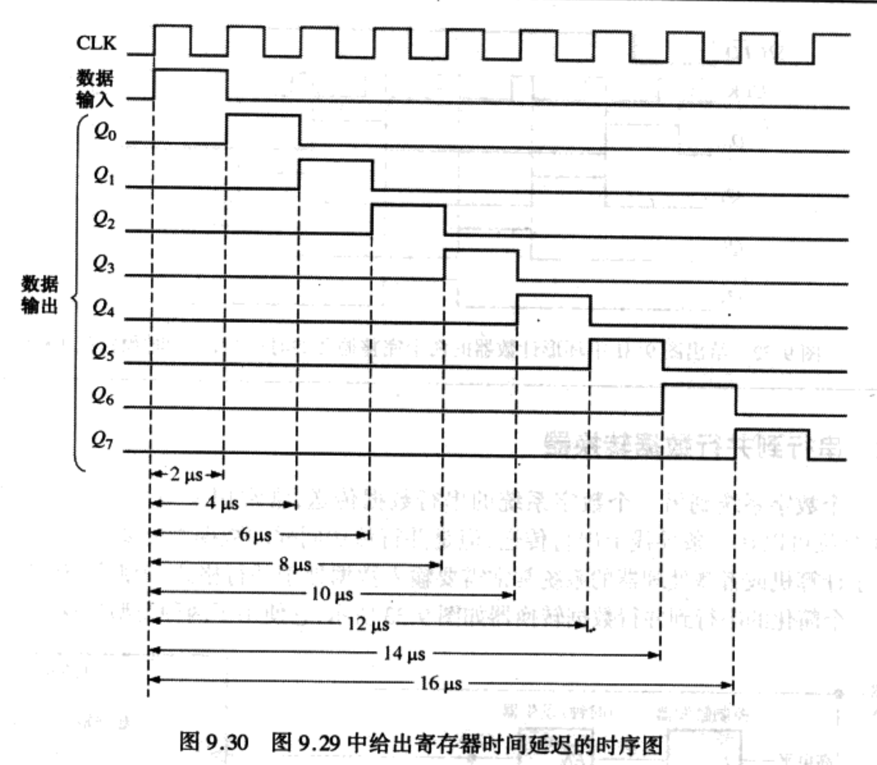


图 9.30　图 9.29 中给出寄存器时间延迟的时序图

使用 74HC195 移位寄存器的环形计数器

如果输出回接到串行输入上，移位寄存器就可以用做环形计数器。图 9.31 用 74HC195 4 位移位寄存器解释了这种应用。

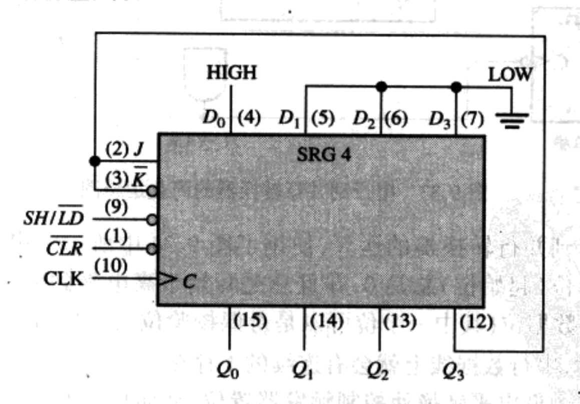


图 9.31　连接为环形计数器的 74HC195

初始时，位组合 1000（或者任何其他组合）可以被同步预置到计数器中，应用此位模式到并行数据输入上，使 $SH/\overline{LD}$ 输入为低电平，同时加上一个时钟脉冲。在这个初始化以后，1 继续在环形计数器中循环，如图 9.32 中的时序图所示。

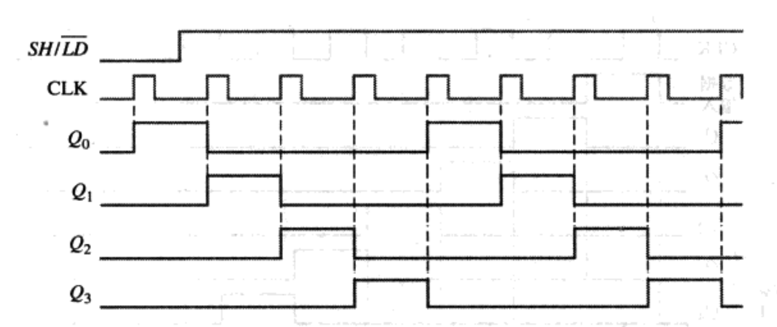


图 9.32 给出图 9.31 中环形计数器的两个完整循环的时序图,初始时预置数 1000

9.8.2 串行到并行数据转换器

从一个数字系统到另一个数字系统的串行数据传送,常常用来减少传输线中的导线数。例如,8 个位可以在一条导线上串行传送,但是并行传送同样的数据则需要 8 条导线。

基于计算机或者微处理器的系统常常需要输入数据处于并行格式,因此需要串行到并行的转换。一个简化的串行到并行数据转换器如图 9.33 所示,它使用了两种类型的移位寄存器。

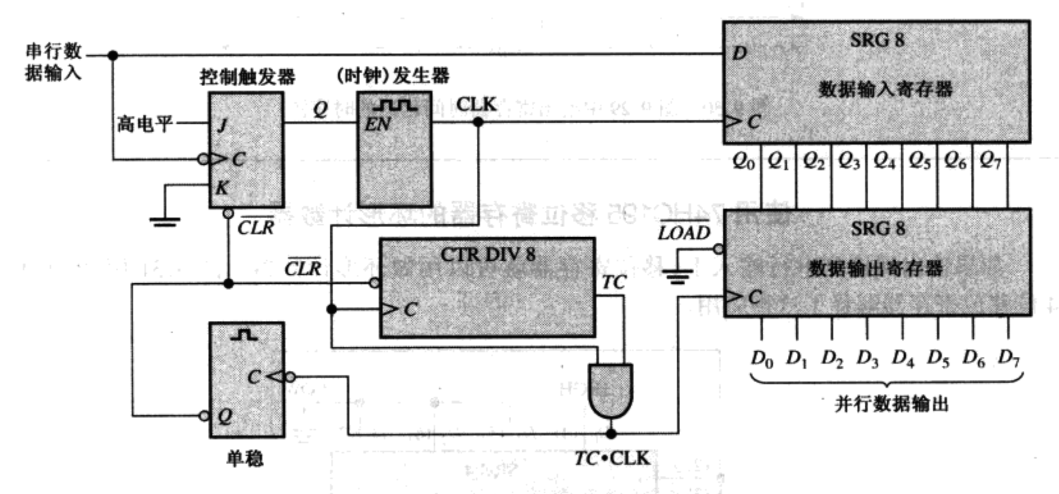


图 9.33 串行到并行转换器的简化逻辑图

为了解释此串行到并行转换器的运算,使用了图 9.34 中所示的串行数据格式。它由 11 个位所组成。第一个位(起始位)总是 0,并且总是开始于高电平到低电平的转换。下面的 8 个位(D_7 到 D_0)都是数据位(其中一个位可以是奇偶校验位),而最后两个位(终止位)总是 1。当没有数据被传送时,串行数据线上就会有连续的 1 存在。

起始位的高电平到低电平转换使控制触发器置位,从而启动时钟发生器。在一个固定的延迟时间之后,时钟发生器开始产生脉冲波形,此波形被应用于数据输入寄存器和 8 分频计数器。时钟的频率精确等于输入串行数据的频率,而起始位之后的第一个时钟脉冲在第一个数据位期间发生。

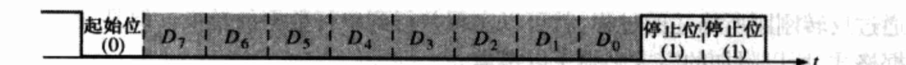


图 9.34 串行数据格式

图 9.35 中的时序图解释了下面的基本运算：8 个数据位（D_7 到 D_0）串行移位到数据输入寄存器。在第 8 个时钟脉冲之后，计数器终端（TC）输出的低电平到高电平的转换，和时钟进行与运算（$TC \cdot \mathrm{CLK}$），从而将数据输入寄存器中的 8 个位载入数据输出寄存器中。与门的输出同时触发单稳（触发输入处的小圆圈去掉）产生一个短期脉冲来清除计数器，并复位控制触发器，从而使时钟发生器停止工作。现在系统已准备好转换下一组 11 位数，并且在起始位的开始处等待下一个高电平到低电平的转换。

图 9.35 图 9.33 中串行到并行数据转换器运算的时序图说明

通过反转刚刚所描述的过程,就可以实现并行到串行数据的转换。但是,由于必须产生串行数据格式,所以附加的需求必须予以考虑。

9.8.3 通用异步接收器和发射机(UART)

正如前面提到的那样,基于计算机和微处理器的系统常常发送和接收并行格式的数据。在许多情况下,这些系统必须和发送和/或者接收串行数据的外部设备进行通信。用来完成这些转换的接口设备是通用异步接收器和发射机(Universal Asynchronous Receiver Transmitter, UART)。图 9.36 解释了基于一般微处理器的系统应用中的 UART。

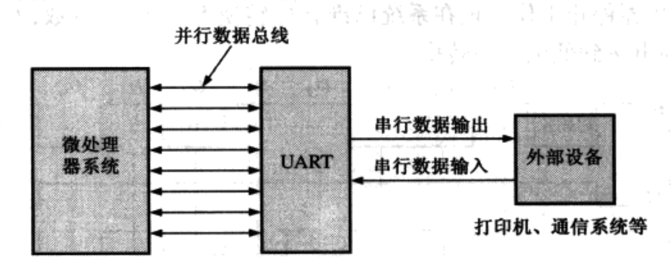


图 9.36　UART 接口

UART 包括一个刚刚讨论的串行到并行数据的转换器和一个并行到串行转换器,如图 9.37所示。数据总线从基本结构上来看是一组并行导线,数据沿着这些导线在 UART 和微处理器系统之间移动。缓冲器连接数据总线和数据寄存器。

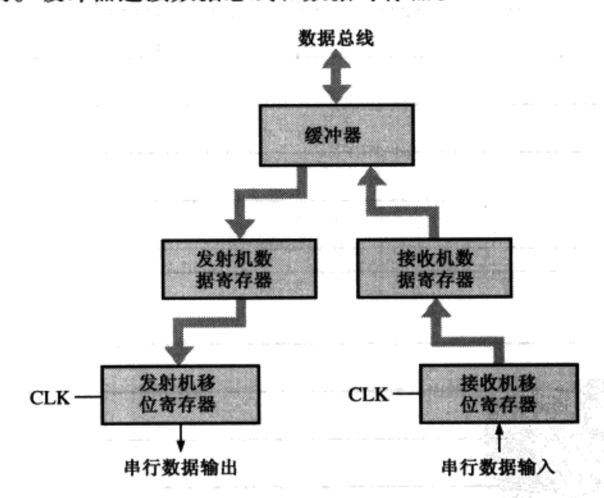


图 9.37　基本 UART 框图

UART 接收串行格式的数据,将数据变换为并行格式,然后将它们放在数据总线上。UART 还接收来自数据总线的并行数据,将这些数据变换为串行格式,然后发送到外围设备上。

9.8.4 键盘译码器

键盘译码器是移位寄存器用做环形计数器,连接于其他设备的一个很好例子。回顾在第 6 章介绍了没有数据存储的简化计算机键盘译码器。

图 9.38 给出了一个简化的键盘译码器，可以对 8 行 × 8 列的 64 键中的一个键进行译码。两个 74HC195 4 位移位寄存器连接为 8 位环形计数器，其中具有 7 个 1 的固定位模式，并且当电源打开时一个 0 会预置到计数器中。两个 74HC147 优先译码器(第 6 章介绍过)被用做 8 线 - 3 线的译码器(9 输入为高电平，8 输出未用)，用来对键盘矩阵中的行(ROW)和列(COLUMN)进行译码。74HC174(十六进制触发器)用做并行输入/并行输出寄存器，其中存储了来自优先译码器的行(ROW)/列(COLUMN)代码。

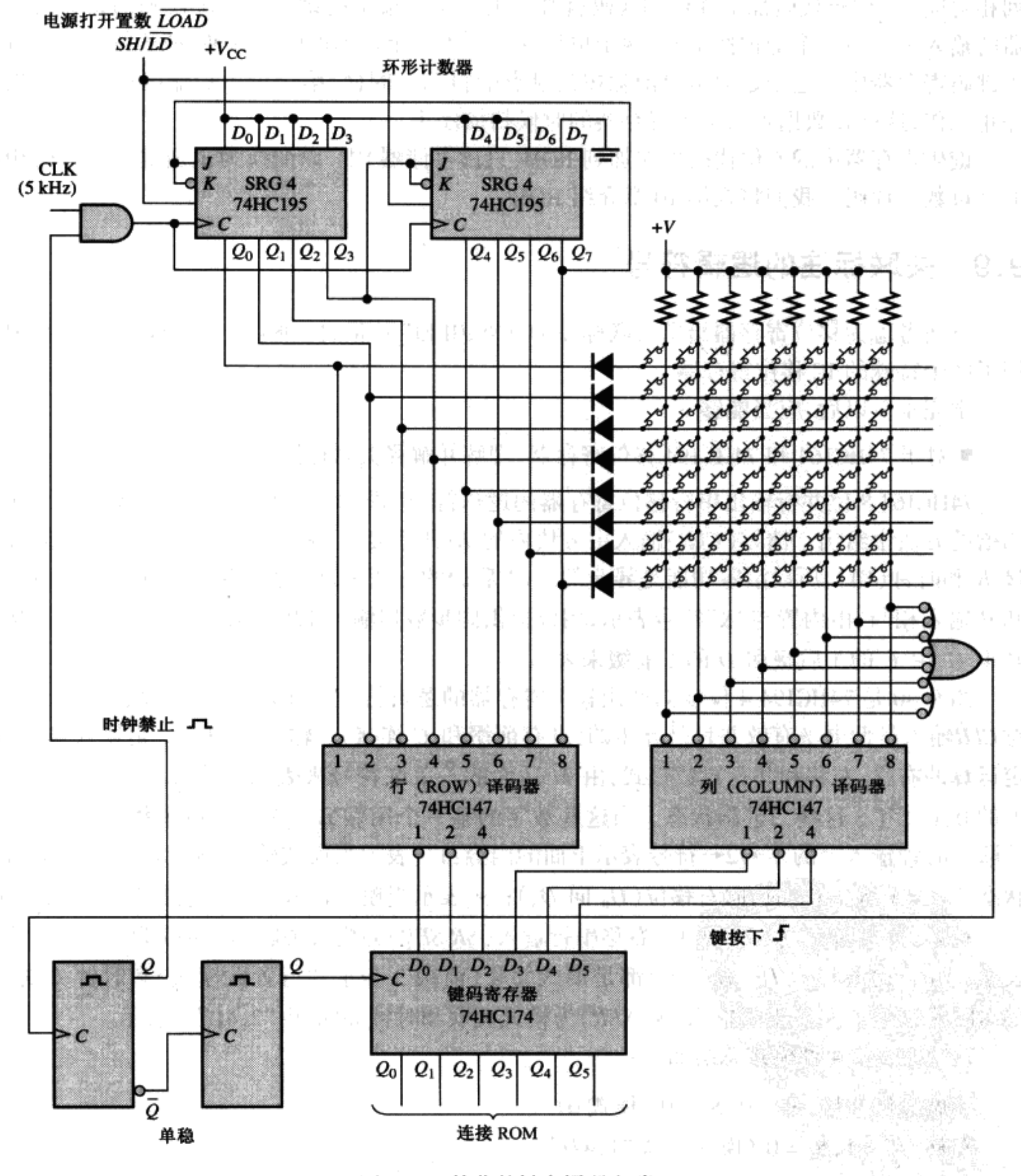


图 9.38　简化的键盘译码电路

图 9.38 中键盘译码器的基本运算如下所示:在 5 kHz 的时钟信号的作用下,0 输出以5 kHz 的速率围绕计数器移位,用来对是否有键按下进行扫描。0(低电平)顺序加在每个行(ROW)线上,同时所有其他的行线都是高电平。所有的行线都连接到行译码器输入上,所以行译码器的 3 位输出在任何时候都是低电平状态下的行线所表示的二进制数。当一个键被按下时,一条列(COLUMN)线就和一条行线连接。当这条行线被环形计数器置为低电平时,那条特殊的列线也同样被置为低电平。行译码器产生一个二进制输出,这个二进制输出和键被按下的那一列相对应。3 位行代码加上 3 位列代码就唯一确定了被按下的键。此 6 位代码加到键码寄存器的输入中。当一个键被按下时,两个单稳就会产生一个延迟的时钟脉冲,把 6 位代码并行置入键码寄存器中。这个延迟可以消除按键触点的抖动。同样,第一个单稳输出使环形计数器停止工作,这样在数据置入键码寄存器的时候扫描停止。

键码寄存器中的 6 位代码现在加到 ROM(只读存储器)中,以变换为识别键盘字符的相应的字母数字代码。我们将在第 10 章介绍 ROM。

9.9 关联标注的逻辑符号

下面考虑为移位寄存器给出关联标注的 ANSI/IEEE 标准 91-1984 符号的两个例子,其中使用了两个特殊的 IC 移位寄存器。

学完本节以后,应当能够

- 对于 74HC164 和 74HC194 移位寄存器,理解并解释关联标注的逻辑符号

74HC164 8 位并行输出串行移位寄存器的逻辑符号如图 9.39 所示。共用的控制输入在有凹槽的方块中给出。清零($\overline{CLR}$)输入由方块内的 R(用于复位)来表示。由于没有关联前缀连接 R 和时钟(C1),所以清零功能是异步的。C1 后面的右箭头符号表示数据从 Q_0 流到 Q_7。A 和 B 输入相与,由内置与“&”符号表示,用以提供同步数据输入 1D 到第一级(Q_0)。注意 D 关联于 C,由 C 的 1 后缀和 D 的 1 前缀来表示。

图 9.40 是 74HC194 4 位双向通用移位寄存器的逻辑符号。开始于控制方块的左上角,注意$\overline{CLR}$输入是低电平有效并且是异步的(没有前缀和 C 连接)。输入 S_0 和 S_1 是模式输入,确定运算的右移、左移和并行置数模式,由 M 后面的$\frac{0}{3}$关联符号来表示。$\frac{0}{3}$表示 S_0 和 S_1 输入上的 0、1、2 和 3 这些二进制状态。当这些数字的某一个用做另一个输入的前缀时,就建立了关联。时钟输入上的 1→/2←符号表示下面的内容:1→表示当模式输入(S_0, S_1)处于二进制 1 状态 ($S_0=1, S_1=0$) 时,向右移位(Q_0 向 Q_3);2←表示当模式输入处于二进制 2 状态($S_0=0$, $S_1=1$)时,向左移位(Q_3 向 Q_0)。右移串行输入(*SR SER*)为模式关联和时钟关联,由“1,4D”来表示。并行输入(D_0, D_1, D_2, D_3)都是模式关联(前缀 3 表示并行置数模式)和时钟关联,由“3,4D”来表示。左移串行输入(*SL SER*)为模式关联和时钟关联,由“2,4D”来表示。

74HC194 的 4 种模式总结如下:

不做任何事情:$S_0=0, S_1=0$ (模式 0)

右移:$S_0=1, S_1=0$ (模式 1,如“1,4D”)

向左移位:$S_0=0, S_1=1$ (模式 2,如“2,4D”)

并行置数:$S_0=1, S_1=1$(模式 3,如“3,4D”)

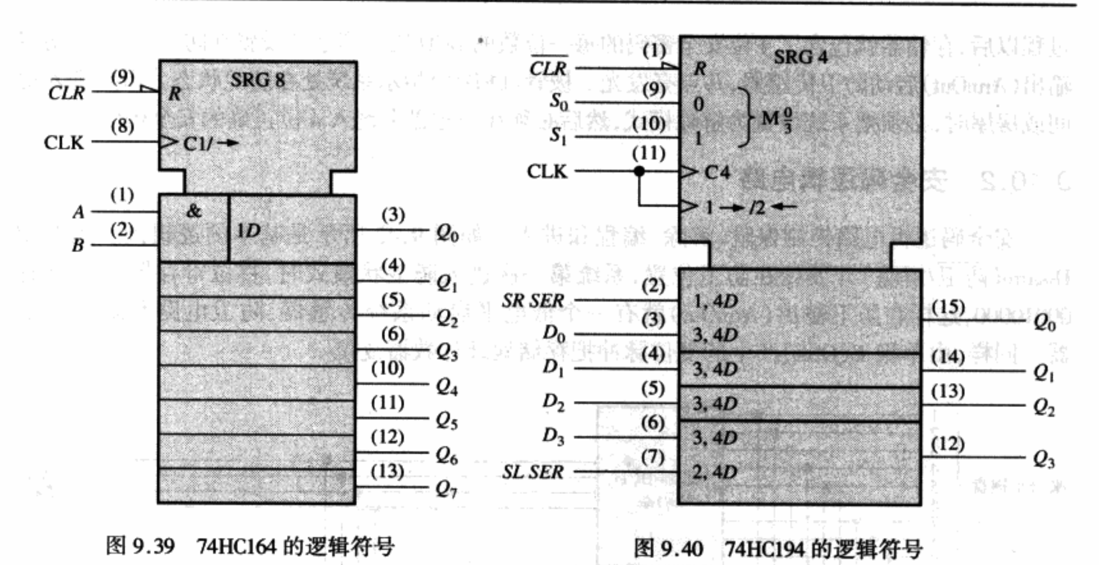


图 9.39　74HC164 的逻辑符号

图 9.40　74HC194 的逻辑符号

9.10　数字系统应用

在这个数字系统应用中,将要设计一个相对简单的房间或房屋的安全控制系统。在解除(防卫)模式下,通过一次性的从小键盘键上得到的 4 位数的安全密码,系统就可以对此码进行编程。一旦输入了安全密码并且存储起来,系统就进入防卫模式。要解除防卫时,就必须在小键盘上键入正确的 4 位码。

9.10.1　基本运算

系统的基本框图如图 9.41 所示。系统由安全密码逻辑电路和存储逻辑单元组成,重点是代码输入逻辑。存储逻辑将在第 10 章介绍,这两个逻辑部分组合在一起形成完整的系统逻辑。

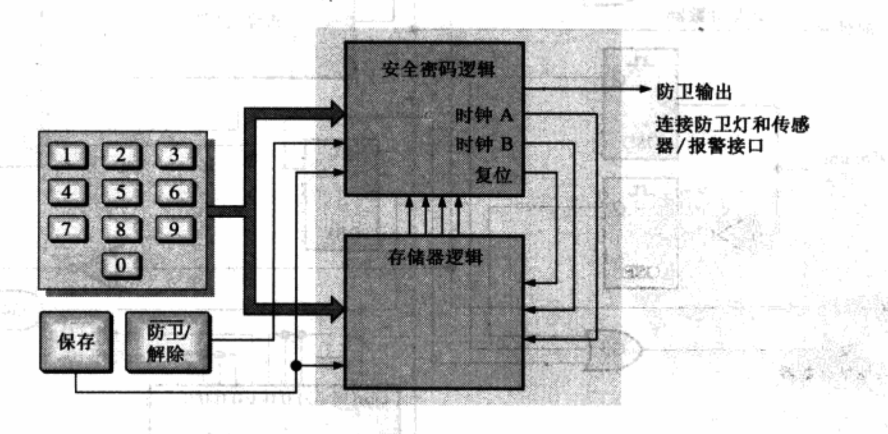


图 9.41　安全系统的基本框图

安全开关$\overline{\text{Arm}}$/Disarm($\overline{\text{防卫}}$/解除)设置系统的防卫或解除模式。首先,把系统设置为解除模式,然后按下安全存储开关,紧接着按下 4 位数字每一位的数字键,这样就完成了编程。在这个

过程以后,存储器就包含了4位安全密码的每一位数的BCD码。当系统设置在防卫模式时,防卫输出(ArmOut)启动防卫传感器,并点亮发光二极管(LED)以指示系统处在防卫状态。需要进入房间或房屋时,必须把系统设置为解除模式,然后必须在小键盘上键入4位正确的安全密码。

9.10.2 安全码逻辑电路

安全码逻辑电路控制保险、解除、编程和进入。如图9.42所示是基本的逻辑图。当$\overline{\text{Arm}}$/Disarm($\overline{\text{防卫}}$/解除)开关处在防卫位置,系统第一次进入防卫状模式时,移位寄存器C包含有00010000,这样在防卫输出(ArmOut)就有一个低电平启动系统传感器、防卫电路和防卫指示器。同样,由单稳E(OSE)产生的复位脉冲把存储地址计数器复位。

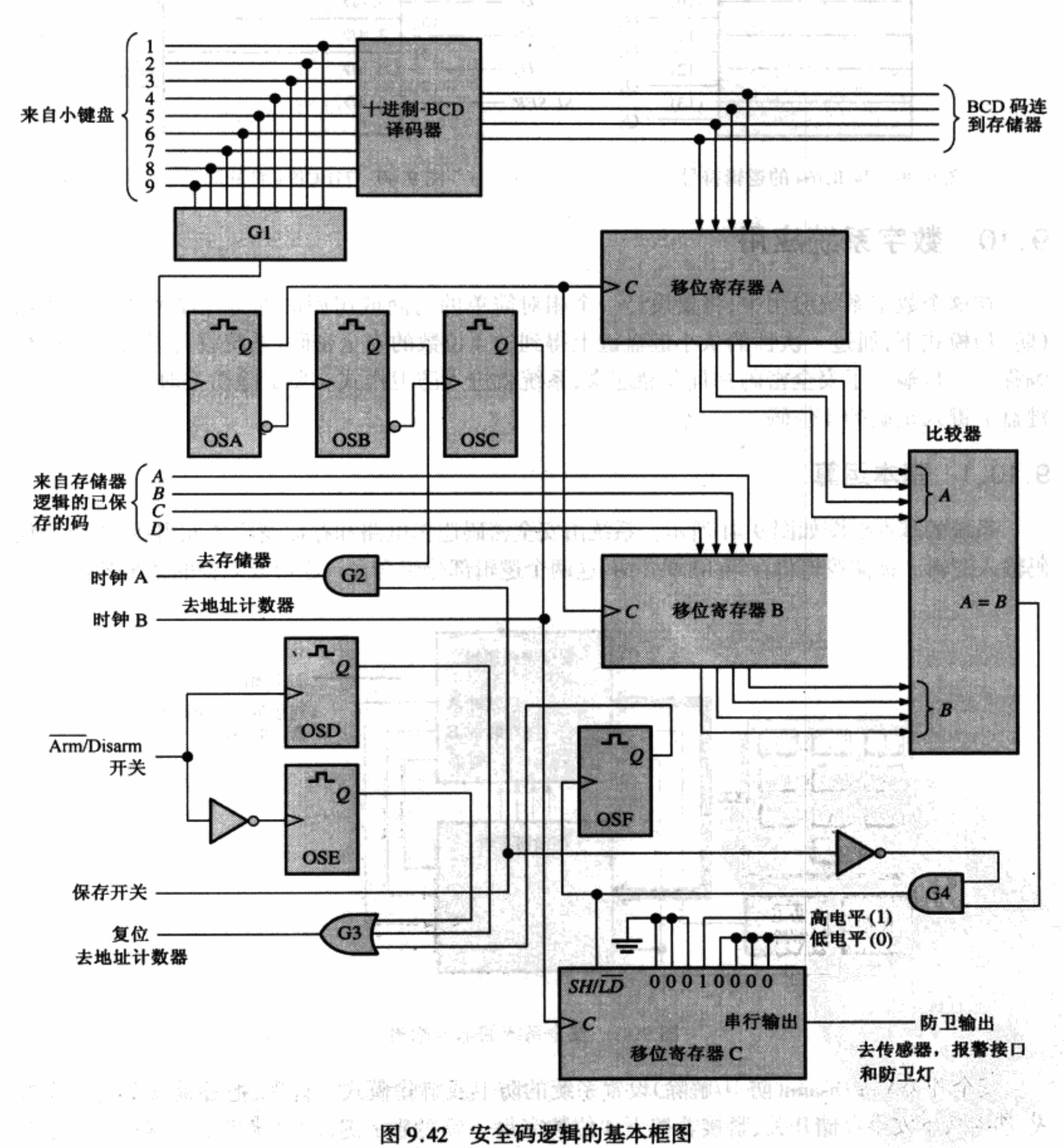


图9.42 安全码逻辑的基本框图

进入 为了使系统解除防卫以便进入安全区域,必须输入和存储在存储器中的码相匹配的正确的4位BCD码。安全密码的第一个数字来自小键盘的输入。十进制-BCD码译码器产生和小键盘上按下的数字相对应的BCD码。单稳A(OSA)由门G1触发产生一个脉冲,使得来自译码器的4位BCD码进入寄存器A,并且来自第一次存储器地址保存的密码进入寄存器B。一旦这些码进入寄存器A和B,就将它们加到比较器的输入。如果在小键盘上键入了正确的数字,比较器A输入的4位数和比较器B输入的4位数相同,那么结果是比较器的 $A=B$ 输出为高电平,并且使得移位寄存器C进入移位(SH)模式。

单稳A输出脉冲的下降沿触发单稳B,单稳B依次又在输出脉冲的下降沿触发单稳C。单稳C的输出为存储地址计数器提供了时钟B,并且使得时钟移位寄存器C的00010000向右移位,结果寄存器中包含00001000。因为串行防卫输出仍然是0(低电平),所以系统保持防卫状态。

当在小键盘上键入第二个正确的数码以后,移位寄存器C中的内容又移位,得到00000100,系统仍然保持防卫状态。当在小键盘上键入第三个数码以后,移位寄存器C中的内容又移位,得到00000010。当在小键盘上键入第四个即最后一个数码以后,移位寄存器C中的内容又移位,得到00000001。这时,串行防卫输出为1(高电平),系统解除防卫允许进入。

如果任何时间键入了不正确的码,比较器的输出为低电平,在移位/$\overline{置数}$($SH/\overline{LD}$)端产生一个低电平,触发单稳F以送出一个复位脉冲到存储地址计数器。这时移位寄存器C处在并行置数模式。这样单稳C触发此寄存器C,随后置入预先连接的码00010000到寄存器。在这种情况下,必须重新键入完整的4位数码。

编程 为4位数码编程输入系统,防卫/解除($\overline{\text{Arm}}$/Disarm)开关处在解除位置。这时触发单稳 D,单稳 D 发送一个复位脉冲通过 $G3$ 传到存储器地址计数器,计数器复位00,即存储器中的首地址。保存开关处在保存位置,使得比较器的输出 $A=B$ 无法通过门 $G4$,而使得单稳 B 的输出经由 $G2$ 提供存储器代码编程期间的时钟。

下一步,所需要的安全密码的第一位通过小键盘输入。作为按键的一个结果,单稳由 G_1 过来的电平触发,依次再触发单稳B,单稳A产生的时钟A把密码保存在存储器中。单稳B触发单稳C,单稳B产生的时钟B作为存储器地址计数器的时钟,使得计数器进入第二个地址(01)。从键盘键入代码的第二位数字,第一位数字所描述的顺序在此重复。键入第四个密码,即最后一个,存储器包含了4位安全密码。如果意外键入了一个错误的数字,这时必须完成4位数的键入或重新开启保存开关,以确保存储器计数器再次包含初始的地址。一旦执行了编程,系统进入防卫模式。

9.10.3 系统安排

- **工作1**:描述移位寄存器A的用途;
- **工作2**:描述移位寄存器B的用途;
- **工作3**:描述移位寄存器C的用途;
- **工作4**:描述比较器的用途;
- **可选工作**:使用74××逻辑集成电路(ICS)和其他元件,完成图9.42中1的安全码逻辑。进行必要的改动,使芯片适合所应用的需求。调试和测试逻辑,并描述设计的流程图(如果有)。

自测题 (答案在本章的结尾)

1. 移位寄存器中的一级由下面哪一项组成?
   (a)锁存器 (b)触发器 (c)存储字节 (d)4 位存储
2. 为了将一字节数据串行移位到移位寄存器中,必须有
   (a)1 个时钟脉冲 (b)1 个置数脉冲
   (c)8 个时钟脉冲 (d)数据中的每个 1 都要有 1 个时钟脉冲
3. 为了将一字节数据并行载入到一个具有同步载入的移位寄存器中,必须有
   (a)1 个时钟脉冲 (b)数据中的每个 1 都要有 1 个时钟脉冲
   (c)8 个时钟脉冲 (d)数据中的每个 0 都要有 1 个时钟脉冲
4. 一组数位 10110101 串行移位(首先移动最右边的位)到一个 8 位并行输出移位寄存器中,其初始状态为 11100100。在两个时钟脉冲之后,该寄存器状态为
   (a)01011110 (b)10110101 (c)01111001 (d)00101101
5. 使用 100 kHz 的时钟频率,8 个数位可以在多少时间内串行进入移位寄存器中?
   (a)80 ms (b)8 ms (c)80 ms (d)10 ms
6. 利用 1 MHz 的时钟频率,8 个数位可以在多少时间内并行进入移位寄存器中?
   (a)在 8 μs 内 (b)在 8 个触发器的传输延迟时间内
   (c)在 1 μs 内 (d)在 1 个触发器的传输延迟时间内
7. 模 10 约翰逊计数器需要
   (a)10 个触发器 (b)4 个触发器
   (c)5 个触发器 (d)12 个触发器
8. 模 10 环形计数器至少需要
   (a)10 个触发器 (b)5 个触发器
   (c)4 个触发器 (d)12 个触发器
9. 当一个 8 位串行输入/串行输出移位寄存器用做 24 μs 的时间延迟时,时钟频率必须是
   (a)41.67 kHz (b)333 kHz (c)125 kHz (d)8 MHz
10. 图 9.38 的键盘译码电路中的环形计数器的目的是
    (a) 顺序为每一行加上高电平,以检测键是否按下
    (b) 为键代码寄存器提供触发脉冲
    (c) 顺序为每一行加上低电平,以检测键是否按下
    (d) 顺序反偏压每一行中的二极管

习题

9.1 节 基本移位寄存器的功能

1. 为什么移位寄存器被认为是基本存储设备?
2. 可以保持两字节数据的寄存器的存储容量是多少?

9.2 节 串行输入/串行输出移位寄存器

3. 对于图 9.43 中的数据输入和时钟,确定图 9.3 中移位寄存器的每一个触发器的状态,并给出 Q 波形。假设该寄存器初始状态为全 1。

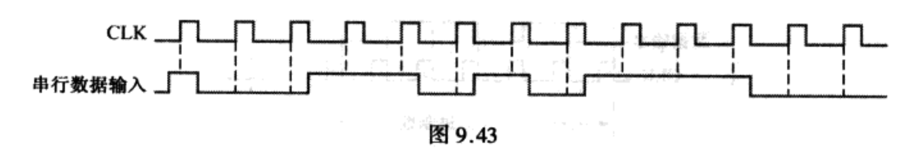


图 9.43

4. 对于图 9.44 中的波形，解出习题 3。

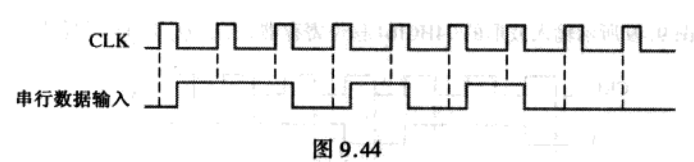


图 9.44

5. 如果图 9.45 中寄存器开始于 101001111000 状态，那么其在每个时钟脉冲之后的状态是什么？

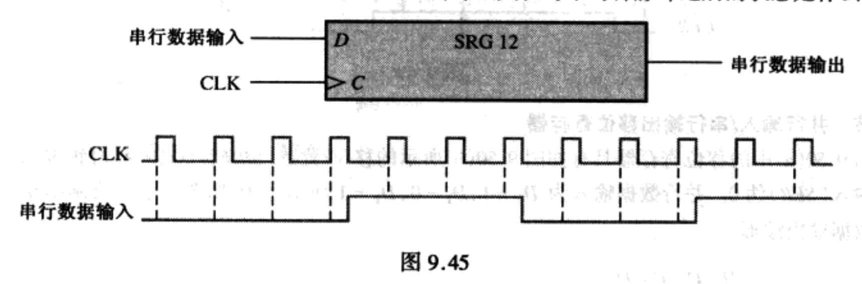
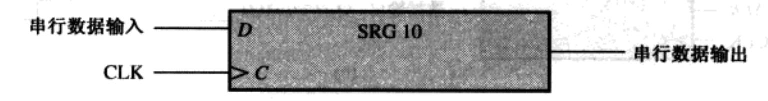


图 9.45

6. 对于串行输入/串行输出移位寄存器，确定图 9.46 中数据输入和时钟波形的数据输出波形。假设该寄存器初始时为清零状态。



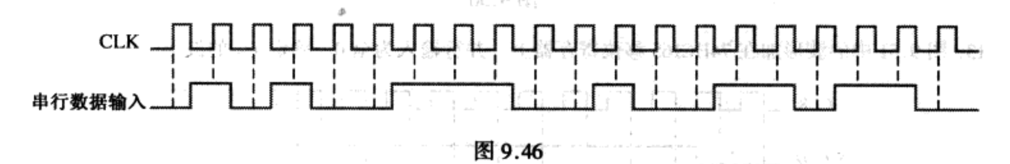


图 9.46

7. 对于图 9.47 中的波形，解出习题 6。

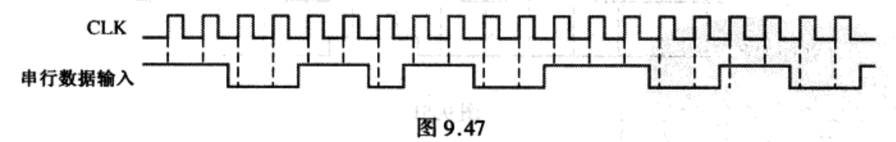


图 9.47

8. 一个时钟上升沿触发的串行输入/串行输出移位寄存器具有如图 9.48 所示的数据输出波形。如果第一个数据位（最左面）是 LSB，存储在 8 位寄存器中的二进制数是多少？

9.3 节　串行输入/并行输出移位寄存器

9. 给出一个完整的时序图，以表示图 9.8 中移位寄存器的并行输出。使用图 9.46 中的波形，寄存器初始时为清零。

10. 对于图 9.47 中的输入波形，解出习题 9。

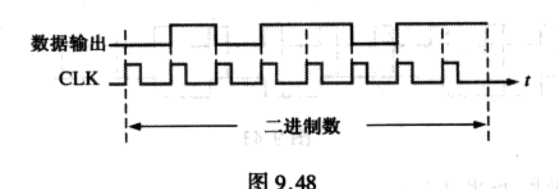


图 9.48

11. 为具有如图 9.49 所示输入波形的 74HC164 移位寄存器,绘制 Q_0 到 Q_7 的输出。

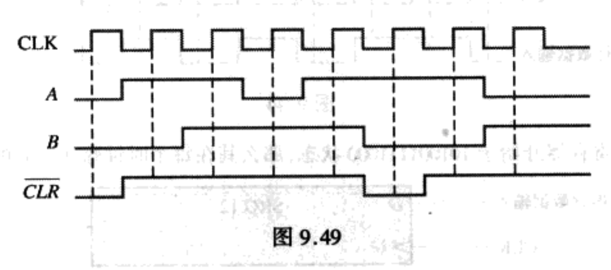


图 9.49

9.4 节 并行输入/串行输出移位寄存器

12. 图 9.50(a)中的移位寄存器具有如图 9.50(b)所示的移位/$\overline{置数}$($SHIFT/\overline{LOAD}$)和时钟输入。串行数据输入(SER)伪 0。并行数据输入为 $D_0=1$、$D_1=0$、$D_2=1$ 和 $D_3=0$,如图所示。绘制出相关于输入的数据输出波形。

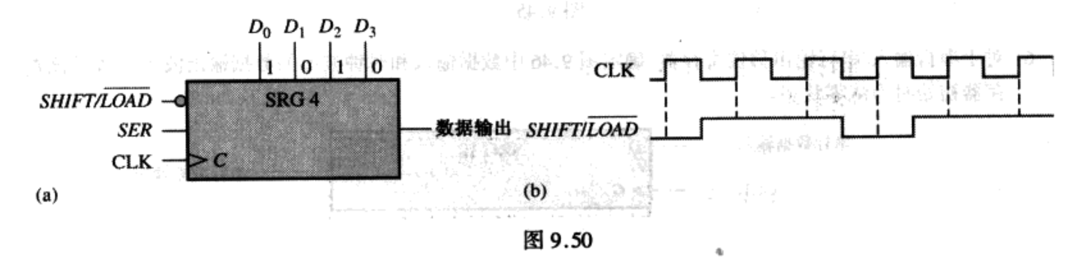


图 9.50

13. 图 9.51 中的波形加在 74HC165 移位寄存器上。并行输入为全 0。确定 Q_7 的波形。

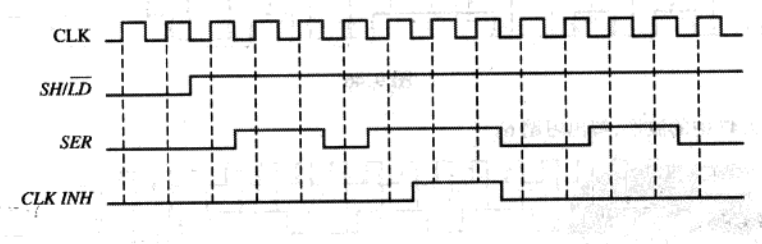


图 9.51

14. 如果并行输入为全 1,解出习题 13。
15. 如果 SER 输入反相,解出习题 13。

9.5 节 并行输入/并行输出移位寄存器

16. 当输入如图 9.52 所示时,确定 74HC195 4 位移位寄存器的所有 Q 输出波形。
17. 如果 $SH/\overline{LD}$ 输入反相,而寄存器初始时清零,解出习题 16。
18. 使用两个 74HC195 移位寄存器来形成一个 8 位移位寄存器。给出所需要的连接。

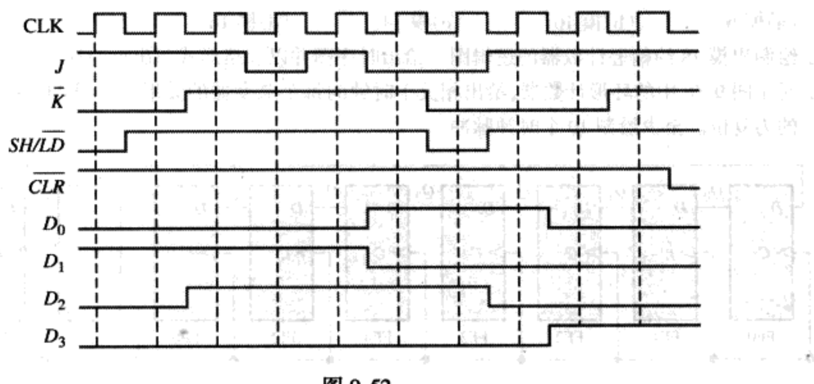


图 9.52

9.6 节　双向移位寄存器

19. 对于图 9.53 中的 8 位双向寄存器，给定了 *RIGHT/LFFT*(右移/左移)控制波形，确定寄存器在每个时钟脉冲之后的状态。这个输入上的高电平启动向右移位，而低电平启动向左移位。假设该寄存器初始时存储了二进制形式的十进制数 76，并且最右边的位置为 LSB。数据输入线上有一个低电平。

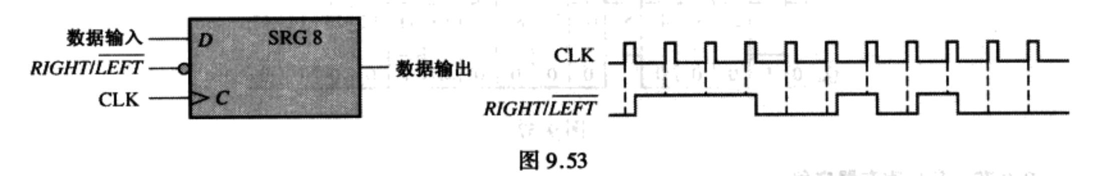


图 9.53

20. 对于图 9.54 中的波形，解出习题 19。

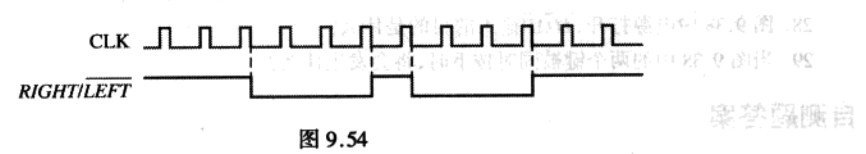


图 9.54

21. 使用两个 74HC194 4 位双向移位寄存器来创建一个 8 位双向移位寄存器。给出连接。

22. 确定具有如图 9.55 所示输入的 74HC194 的 Q 输出。输入 D_0、D_1、D_2 和 D_3 都是高电平。

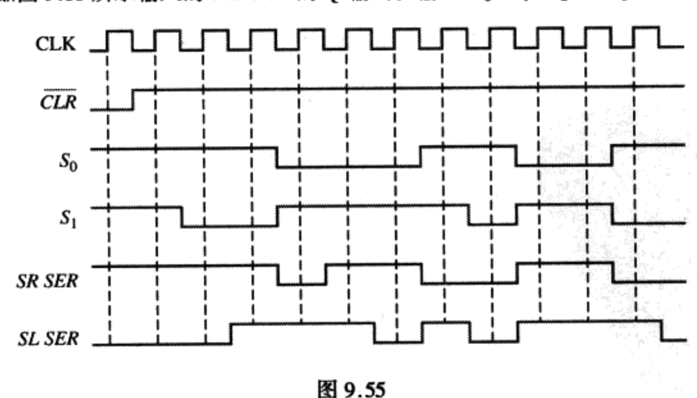


图 9.55

9.7 节　移位寄存器计数器

23. 实现下面每一个约翰逊计数器配置分别需要多少个触发器：

(a)模 6　　(b)模 10　　(c)模 14　　(d)模 16

24. 绘制出模 18 约输逊计数器的逻辑图。给出时序图并以表格形式写出该时序。

25. 对于图 9.56 中的环形计数器，给出相关于时钟的每个触发器的波形。假设 FF0 初始时为置位，其余的为复位。至少绘制 10 个时钟脉冲。

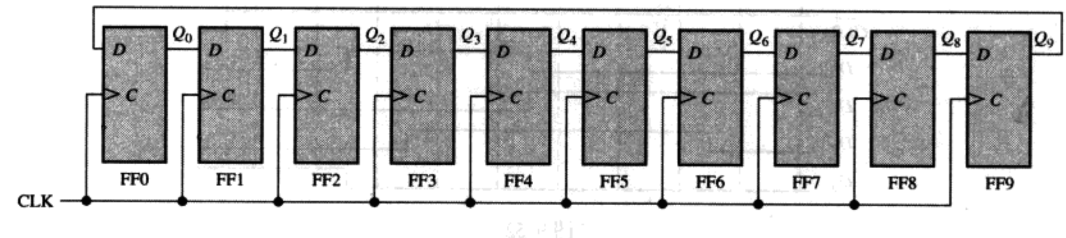


图 9.56

26. 需要如图 9.57 所示的波形模式。设计一个环形计数器，并指出怎样进行预置才能在它的 Q_9 输出上产生这个波形。在 CLK16 处，该模式开始重复。

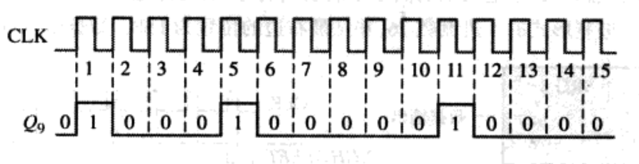


图 9.57

9.8 节　移位寄存器应用

27. 使用 74HC195 4 位移位寄存器来实现一个 16 位环形计数器。给出连接。

28. 图 9.38 中电源打开，$\overline{LOAD}$输入的目的是什么？

29. 当图 9.38 中的两个键被同时按下时，将会发生什么？

自测题答案

1. (b)　2. (c)　3. (a)　4. (c)　5. (a)　6. (d)　7. (c)　8. (a)　9. (b)　10. (c)

第10章　内存和外存

章节提纲

10.1　半导体存储器基础

存储器是系统中大量存储二进制数据的部分。半导体存储器由存储元件阵列组成,这些存储元件通常是锁存器或者电容。

学完本节以后,应当能够

- 解释存储器怎样存储二进制数据
- 讨论存储器的基本组织结构
- 描述写操作
- 描述读操作
- 描述寻址操作
- 解释什么是 RAM 和 ROM



计算机小知识

字的一般定义是,由二进制数据的一个单位组成的完整信息单位。当应用于计算机指令时,字就很明确地定义为两个字节(16 位)。作为在计算机中使用的汇编语言的重要组成部分,DW(定义字)指令的意思是以 16 位单元来定义数据。这个定义独立于特殊的微处理器或者数据总线的长度。汇编语言还允许用 DB 指令来定义字节(8 位),用 DD 指令来定义双字(32 位),用 QD 指令来定义四字(64 位)。

10.1.1　二进制数据的单位:位、字节、半字节和字

作为一种规则,存储器以单位存储数据,这些单位从 1 位到 8 位。正如我们所知道的那样,二进制数据的最小单位是位。在许多应用中,以 8 位一个单元处理数据,或者以多个 8 位单元处理数据,这种 8 位单元称为字节。一个字节可以分为两个 4 位单元,称为半字节。完整的信息单位称为字,一般由一个或者多个字节组成。有些存储器以 9 位数组存储数据;9 位数组由一个字节加上一个奇偶校验位组成。

10.1.2 基本半导体存储阵列

存储器中的每一个存储元件都可以保存一个 1 或者 0,它们被称为单元。存储器由单元阵列组成,如图 10.1 所示,示例中使用了 64 个单元。存储阵列中的每个方块表示一个存储单元,并且它的位置通过指定行和列来确定。

64 单元阵列可以基于数据单位,以几种不同的方式来组织。图 10.1(a)给出了一个 8×8 阵列,可以将其看做 64 位内存或者 8 字节内存。图 10.1(b)给出了一个 16×4 阵列,它是一个 16 个半字节内存,而图 10.1(c)给出了一个 64×1 阵列,它是一个 64 位内存。内存由其可以存储的字数乘以字长来指定。例如,16k×8 的内存可以存储 16 384 个 8 位字。内存术语的矛盾在这里是很常见的。字的实际个数总是 2 的幂,即在这种情况下是 2^{14} = 16 384;但是,通常的做法是以最接近千的数来表示这个数目,在这里是 16k。

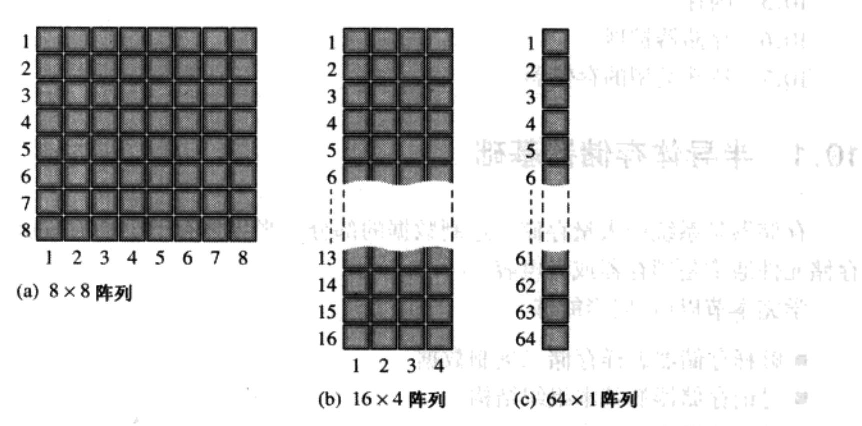


图 10.1 以三种不同方式组成的 64 单元内存

10.1.3 存储器地址和容量

存储阵列中数据单元的位置称为它的地址。例如,在图 10.2(a)中,阵列中某位的地址由如图所示的二维阵列的行和列来指定。在图 10.2(b)中,字节的地址只由行来指定。所以,正如所看到的那样,地址取决于存储器的数据单位是如何组成的。个人计算机具有以字节组成的随机存储器。这就意味着可以寻址的最小位组为 8 位。

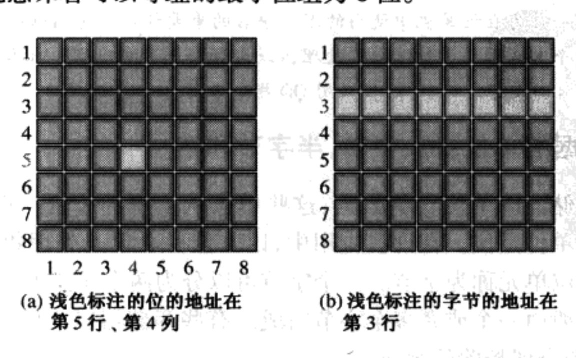
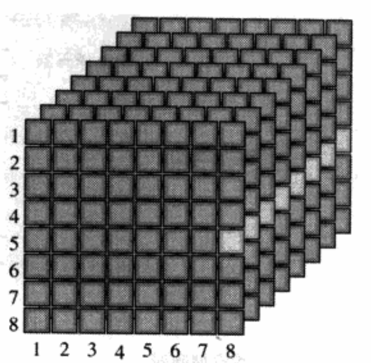


图 10.2 二维阵列存储器地址的例子

在图 10.3 中，一个三维阵列中的字节的地址如图所示，由行和列来指定。在此情况下，可以访问的最小位组是 8 位。存储器的容量是存储器可以存储的数据单位总数。例如，对于图 10.2(a)中以位组成的存储阵列，容量是 64 位。对于图 10.2(b)中以字节组织的存储阵列，容量是 8 字节，也是 64 位。在图 10.3中，容量是 64 字节。计算机存储器一般具有 256 MB(兆字节)或者更多的内存。



浅色标注的字节的地址在第 5 行、第 8 列

图 10.3 三维阵列中存储器地址的例子

10.1.4 存储器的基本操作

由于存储器存储二进制数据，所以必须把数据放到存储器中，并且在需要时从内存中复制数据。写操作将数据放到存储器中指定的地址，读操作把存储器中指定地址的数据复制出来。寻址操作是写操作和读操作的共同部分，用以选择指定的存储器地址。

数据单位在写操作时进入内存，而在读操作时离开内存，这都要经过称为数据总线的一系列线。如图 10.4 所示，数据总线是双向的，也就意味着数据的传送可以有两个方向(进入存储器或者离开存储器)。在这个以字节组成的存储器中，数据总线至少有 8 条线，这样使得选中地址中的 8 位数以并行的方式传送。对于写或者读操作来说，通过把所需地址表示的二进制码放到称为地址总线的一组线上来选择地址。地址码由内部译码，随后选择正确的地址。对于图 10.4(b)中的三维存储阵列的情况，有两个译码器，一个用于行译码，一个用于列译码。总线上线的数目由存储器的容量确定。例如，一个 15 位地址码可以选择存储器中的 32 768 个位置(2^{15})，一个 16 位的地址码可以选择存储器中的 65 536 个位置(2^{16})，以此类推。在个人计算机中，32 位的地址总线可以选择 4 294 967 296 个位置(2^{32})，表示为 4 G。

写操作 图 10.5 给出了一个简化的写操作。为了在存储器中保存一个字节的数据，地址寄存器所保存的代码被放到地址总线上。一旦地址码位于地址总线上，地址译码器就会对地址译码并且在内存中选择指定的位置。存储器随后得到一个写命令，数据寄存器中的数据字节就被放到数据总线上，并且存储在指定的存储器地址中，因而完成了写操作。当新的数据字节写入存储器地址时，存储在那个地址上的当前数据字节就会被覆盖(由新的数据字节替代)。

读操作 一个简化的读操作如图 10.6 所示。同样，地址寄存器中的代码放到地址线上。一旦地址码位于总线上，地址译码器就会对地址译码并且选择存储器中指定的位置。存储器随后得到一个读命令，存储在所选择存储地址中的数据字节的一个“副本”被放到数据总线上，并放入数据寄存器，因此完成了读操作。当数据字节从存储器地址中读出时，它仍然保存在那个地址中。这称为非破坏性读出。

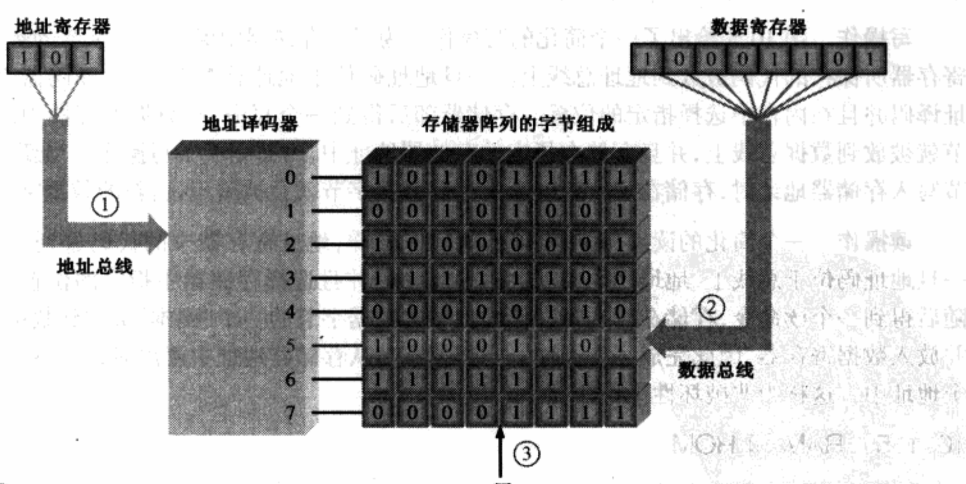
10.1.5 RAM 和 ROM

两类主要的半导体存储器是 RAM 和 ROM。RAM(随机存储器)类型的存储器，在等量时间内可以访问所有的地址，并且可以按照任何顺序来选择这些地址以执行读或者写操作。所有的 RAM 都具有读和写的能力。因为 RAM 在电源关闭时会丢失所存储的数据，所以它是易失性内存。

(a) 二维存储阵列

(b) 三维存储阵列

图 10.4 给出地址总线、地址译码器、双向数据总线和读/写输入的二维和三维存储器的框图



① 地址码 101 放到地址总线上，并选择了地址 5

② 数据字节放到数据总线上

③ 写命令使得数据字节存储在地址 5 中，取代先前的数据

图 10.5 写操作的解释

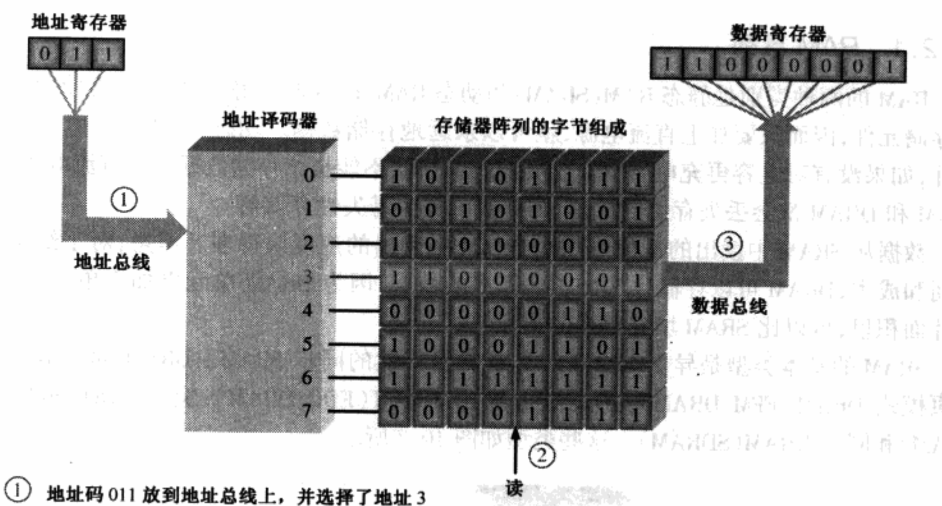


图 10.6　读操作的解释

ROM(只读存储器)类型的存储器是永久或者半永久地保存数据。数据可以从 ROM 中读出,但是和 RAM 不同,它没有写操作。ROM 和 RAM 相似,也是随机存储器,但是术语 RAM 的传统意思是随机读/写存储器。本章将会介绍几种类型的 RAM 和 ROM。因为 ROM 在电源关闭后仍然保存着数据,所以它是永久性存储器。

10.2　随机存储器(RAM)

RAM 是一种读/写存储器,数据可以任意的序列写入或者从任意选择的地址中读出。当数据单元写入 RAM 中给定的地址中时,先前存储在此地址的数据单元就会被新数据单元替代。当数据单位从 RAM 中给定的地址中读出时,此数据单位仍然保存着,并未被读操作擦除。这种非破坏性读操作可以看做对地址内容的复制而不影响地址中的内容。RAM 典型的用途是短期存储数据,因为当电源关闭后,它不能保存存储的数据。

学完本节以后,应该能够

- 给出两种类型的 RAM 的名字
- 解释 SRAM 是什么
- 描述 SRAM 存储单元
- 解释异步 SRAM 和同步突发 SRAM 之间的区别
- 解释什么是 DRAM
- 描述 DRAM 存储单元
- 讨论 DRAM 的类型
- 比较 SRAM 和 DRAM

10.2.1 RAM 系列

RAM 的两种类别是静态 RAM(SRAM)和动态 RAM(DRAM)。静态 RAM 通常使用锁存器作为存储元件,因而只要加上直流电源,就可以永远地存储数据。动态 RAM 使用电容作为存储元件,如果没有对电容再充电,则称之为刷新的过程,不能长期存储数据。当直流电源移走后,SRAM 和 DRAM 都会丢失存储的数据,因此被归类为易失性存储器。

数据从 SRAM 中读出的速度要比从 DRAM 中读出的速度快得多。但是,对于给定的物理空间和成本,DRAM 可以存储比 SRAM 多得多的数据,因为 DRAM 单元更加简单,并在给定的芯片面积里,可以比 SRAM 填充更多的单元。

SRAM 的基本类型是异步 SRAM 和具有突发特性的同步 SRAM。DRAM 的基本类型是快速页模式 DRAM(FPM DRAM)、扩充数据输出 DRAM(EDO DRAM)、突发 EDO DRAM(BEDO DRAM)和同步 DRAM(SDRAM)。这些类型如图 10.7 所示。

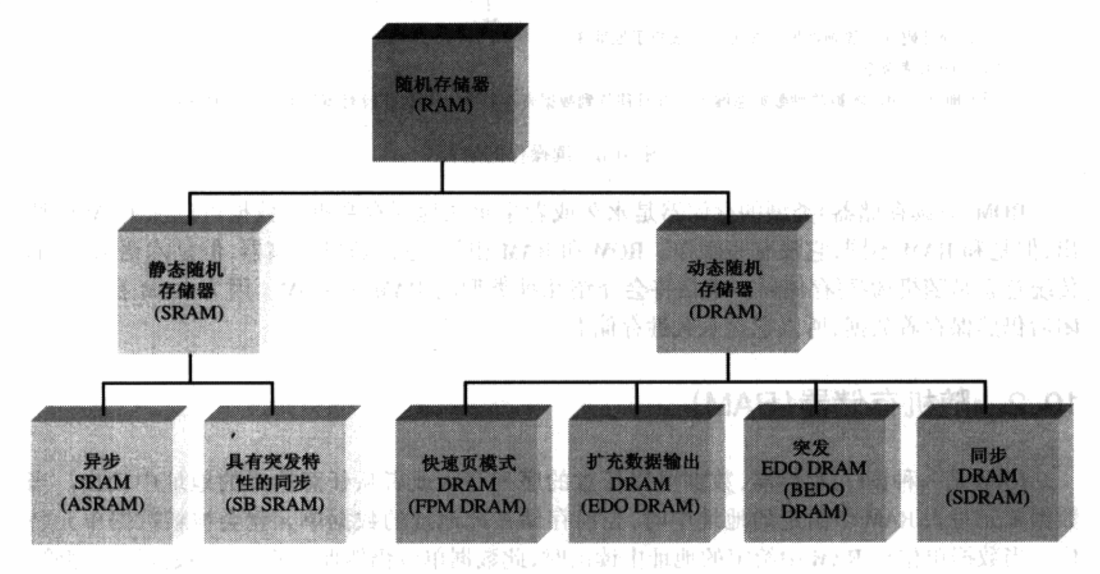


图 10.7 RAM 系列

10.2.2 静态 RAM(SRAM)

存储单元 所有的静态 RAM 的特点是锁存器存储单元。只要直流电源加在 SRAM 上,它就可以永远保持 1 或 0 的状态。如果电源移去,存储的数据位就会丢失。

图 10.8 给出一个基本的 SRAM 锁存器存储单元。这个单元由选择(Select)线上的有效电平所选择,数据位(1 或 0)通过数据写入(Data in)线写入单元。一个数据位由数据读出(Data out)线读出。

基本静态存储单元阵列 SRAM 中的存储单元由行和列组成,如图 10.9 所示为一个 $n\times4$ 阵列的情况。在每行中所有的单元享有相同的行选择(Row Select)线。在数据输出线中的每一组数据进入给定的行中的每个单元,并且连接到一条数据线,作为通过数据输入和数据输出缓冲器的一个数据输入和数据输出。

为了将一个数据单位(这里是半字节)写入到内存阵列中给定的单元行中,行选择(Row Se-

lect)线就会进入它的有效状态,并且 4 个数据位被置于数据输入(Data I/O)线上。这时写入线进入它的有效状态,使得每一个数据位保存到相关列中选中的单元上。为了读出一个数据单元,读出线进入它的有效状态,使得存储在选择行中的 4 个数据位出现在数据输出(Data I/O)线上。

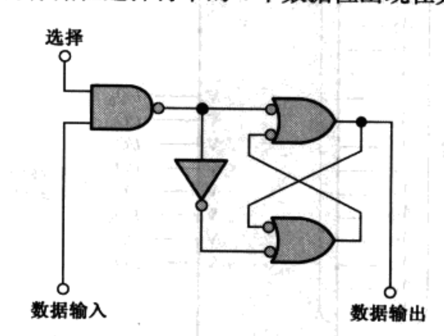


图 10.8 一个典型的 SRAM 锁存器存储单元

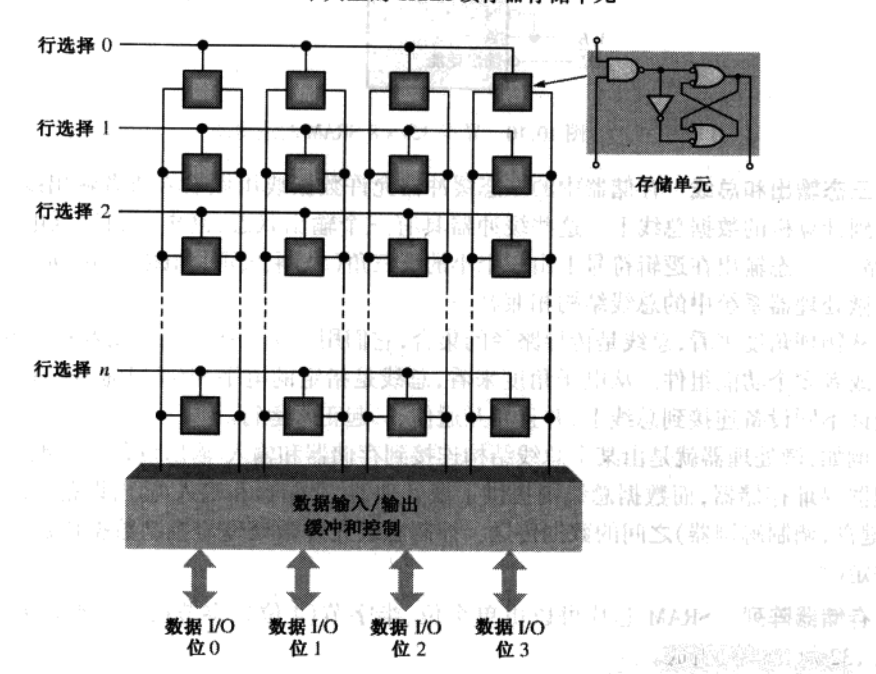


图 10.9 基本 SRAM 阵列

10.2.3 基本异步 SRAM 的组成结构

异步 SRAM 的操作并不同步于系统时钟。为了解释 SRAM 的一般组成结构,使用了一个 32k × 8 位的内存。该内存的逻辑符号如图 10.10 所示。

在读(READ)模式时,存储在所选择地址中的 8 个数据位出现在数据输出线上。在写(WRITE)模式中,加在数据输入线上的 8 个数据位被存储到所选择的地址中。数据输入和数据输出线(I/O_0 到 I/O_7)享用相同的线。在读(READ)期间,它们用做输出线(O_0 到 O_7);在写(WRITE)期间,它们用做输入线(I_0 到 I_7)。

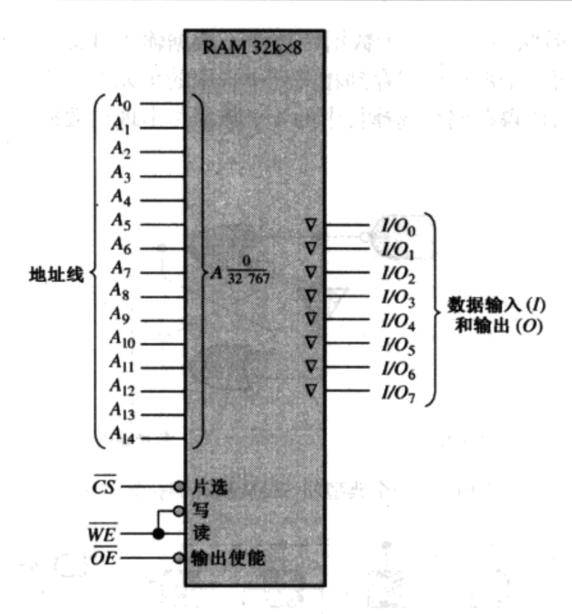


图 10.10　异步 32k × 8 SRAM 的逻辑图

三态输出和总线　存储器中的三态缓冲器允许数据线用做输入或者输出线,并把存储器连接到计算机的数据总线上。这些缓冲器具有三个输出状态:高电平(1)、低电平(0)和高阻(开路)。三态输出在逻辑符号上由一个小的反三角(▽)来表示,如图 10.10 所示,它们可以和基于微处理器系统中的总线结构相兼容。

从物理角度来看,总线是传导路径的集合,它们用于交互连接系统或者几个不同系统中的两个或者多个功能组件。从电子角度来看,总线是指定的电平和/或当前电平和信号的集合,它允许不同设备连接到总线上,并且相互通信、一起正常运行。

例如,微处理器就是由某个总线结构连接到存储器和输入/输出设备上。地址总线允许微处理器寻址存储器,而数据总线则提供了微处理器、存储器和输入/输出设备(如监测器,打印机,键盘,调制解调器)之间的数据传送。控制总线允许微处理器控制数据传送并且为不同的组件定时。

存储器阵列　SRAM 芯片可以由单个位、半字节(4 位)、字节(8 位)或者多字节(16 位,24 位,32 位,等等)组成。

图 10.11 给出了典型 32k × 8 SRAM 的组成结构。存储器单元阵列排列成 256 行和 128 列,每一行和列都具有 8 个位,如图 10.11(a)所示。实际上是 2^{15} = 32 768 个地址,每个地址都具有 8 个位。这个例子的存储容量是 32 768 个字节(一般表示为 32k 字节)。

图 10.11(b)中的 SRAM 工作如下。首先,要使存储器工作,芯片 $\overline{CS}$ 选择必须为低电平。15 条地址线中的 8 条由行译码器译码以选择 256 行中的某一行。15 条地址线中的 7 条由列译码器译码以选择 128 个 8 位列中的某一列。

读(READ)　在读(READ)模式中,写使能输入 $\overline{WE}$ 为高电平,而输出使能 $\overline{OE}$ 为低电平。输入三态缓冲器由门 G_1 禁止,而列输出三态缓冲器由门 G_2 开启。因此,来自所选地址的 8 个数据位途经列 I/O 到达数据线(I/O_0 到 I/O_7),这时(I/O_0 到 I/O_7)作为数据输出线。

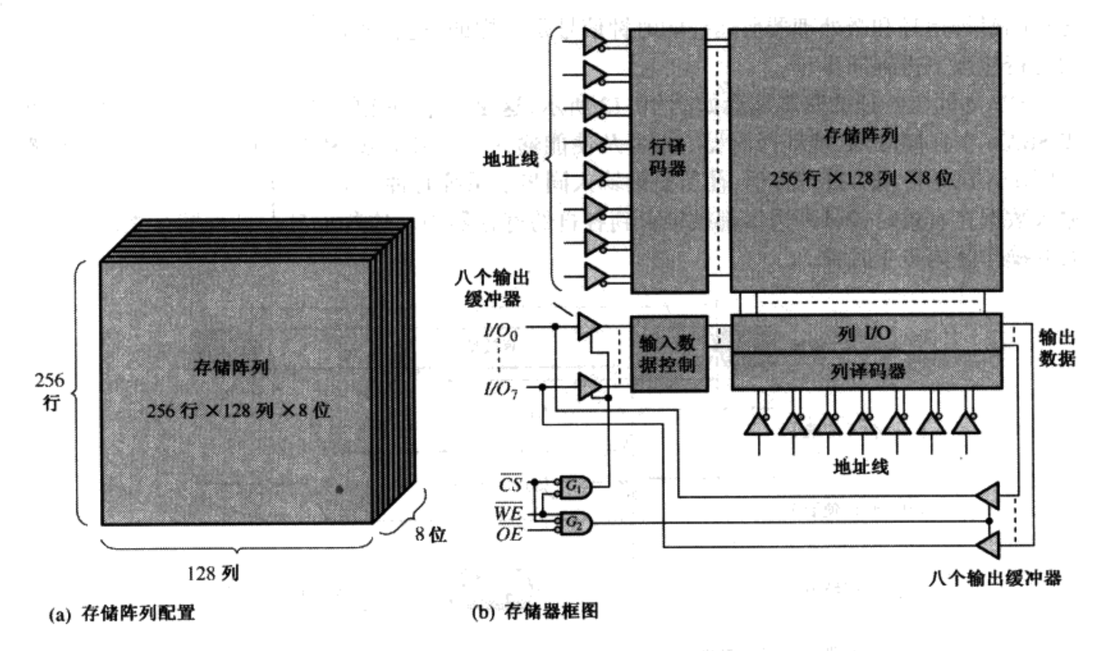


图 10.11 异步 32k × 8 SRAM 的基本组成结构

写(WRITE) 在 WRITE 模式时,$\overline{WE}$为低电平而$\overline{OE}$为高电平。输入缓冲器由门 G_1 开启,而输出缓冲器由门 G_2 禁止。因此,位于数据线上的 8 个输入数据位途经输入数据控制和列 I/O到达所选择的地址中,并被保存。

读写周期 图 10.12 给出了存储器读周期和写周期的典型时序图。对于图 10.12(a)所示的读周期,一个有效地址码在一个特定的时间间隔加在地址线上,这个时间间隔称为读周期 t_{RC}。接下来,片选($\overline{CS}$)和输出使能($\overline{OE}$)输入变为低电平。在$\overline{OE}$输入变为低电平后再经过一个时间间隔,来自所选择地址中的一个有效数据字节就会呈现在数据线上。这个时间间隔称为输出使能存取时间 t_{GQ}。读周期的另外两种存取时间是地址存取时间 t_{AQ}和芯片使能存取时间 t_{EQ}。t_{AQ}从有效地址开始到数据线上出现有效数据为止;t_{EQ}从$\overline{CS}$的高电平到低电平的转换到数据线上出现有效数据为止。

在每个读周期期间,数据的一个单位(这里就是一个字节)就从存储器中读出。

对于图 10.12(b)的写周期,有效地址码在指定的时间间隔里加在地址线上,这个时间间隔称为写周期 t_{WC}。接下来,片选($\overline{CS}$)和写使能($\overline{WE}$)输入变为低电平。从有效地址开始,直至输入$\overline{WE}$变为低电平所需的时间间隔称为地址建立时间 $t_{s(A)}$。而$\overline{WE}$输入必须为低电平的时间是写脉冲宽度。在有效数据加到数据输入后,输入$\overline{WE}$必须保持低电平的时间命名为 t_{WD};而在输入变为高电平之后,有效输入数据必须保持在数据线上的时间为数据保持时间 $t_{h(D)}$。

在每个写周期期间,一个数据单位被写入存储器中。

10.2.4 基本同步突发 SRAM 组成结构

和异步 SRAM 不同,同步 SRAM 和系统时钟同步。例如,在一个计算机系统中,同步 SRAM

运行的时钟信号和微处理器所运行的时钟信号是一样的,这就使得微处理器和存储器在较快的运行速度下得到同步化。

SRAM 同步特征的基本概念如图 10.13 所示,这是一个简化的 32k×8 存储器的框图。同步 SRAM 在存储阵列、地址译码器、读/写及使能输入方面和异步 SRAM 非常相似。基本区别是同步 SRAM 使用时钟寄存器,使所有的输入同步于系统时钟。地址、读/写输入、芯片使能和输入数据在有效时钟脉冲边沿都被锁存到各自的寄存器中。这些信息一旦被锁存之后,存储器的操作就同步于时钟。

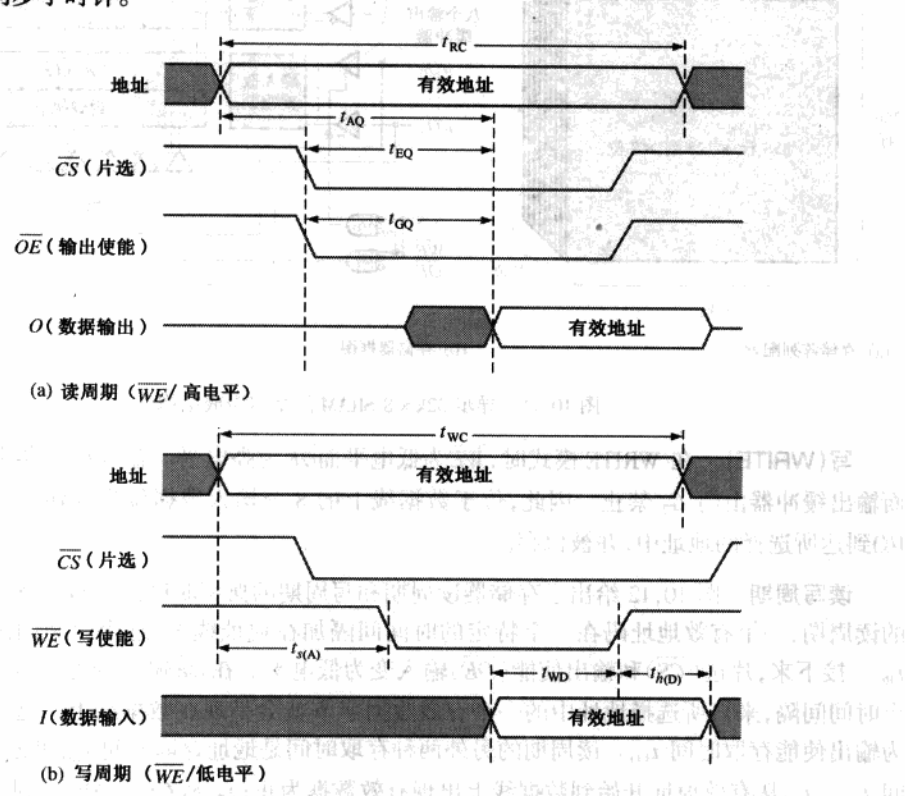


图 10.12 图 10.11 中 SRAM 的基本读写周期的时序

出于简化的目的,图 10.13 引入了多并行线或者总线的符号,作为分别绘制每条线的替换方案。一组并行线可以由一条加粗线表示,加粗线同时还带有一条斜线和独立线的个数。例如,下面的符号表示 8 条并行线的集合:



地址位 A_0 到 A_{14} 在时钟脉冲的上升沿锁存到地址寄存器中。在相同的时钟脉冲作用下,写使能($\overline{WE}$)线和片选($\overline{CS}$)的状态分别锁存到写寄存器和使能寄存器中。这些都是 1 位寄存器或者简单的触发器。同样,在相同的时钟脉冲作用下,对于写操作,输入数据锁存到数据输入寄存器中;对于读操作,选中的存储器地址中的数据锁存到数据输出寄存器中,这都由基于输入的数据 I/O 控制来确定。这些输入来自写寄存器、使能寄存器和输出使能($\overline{OE}$)。

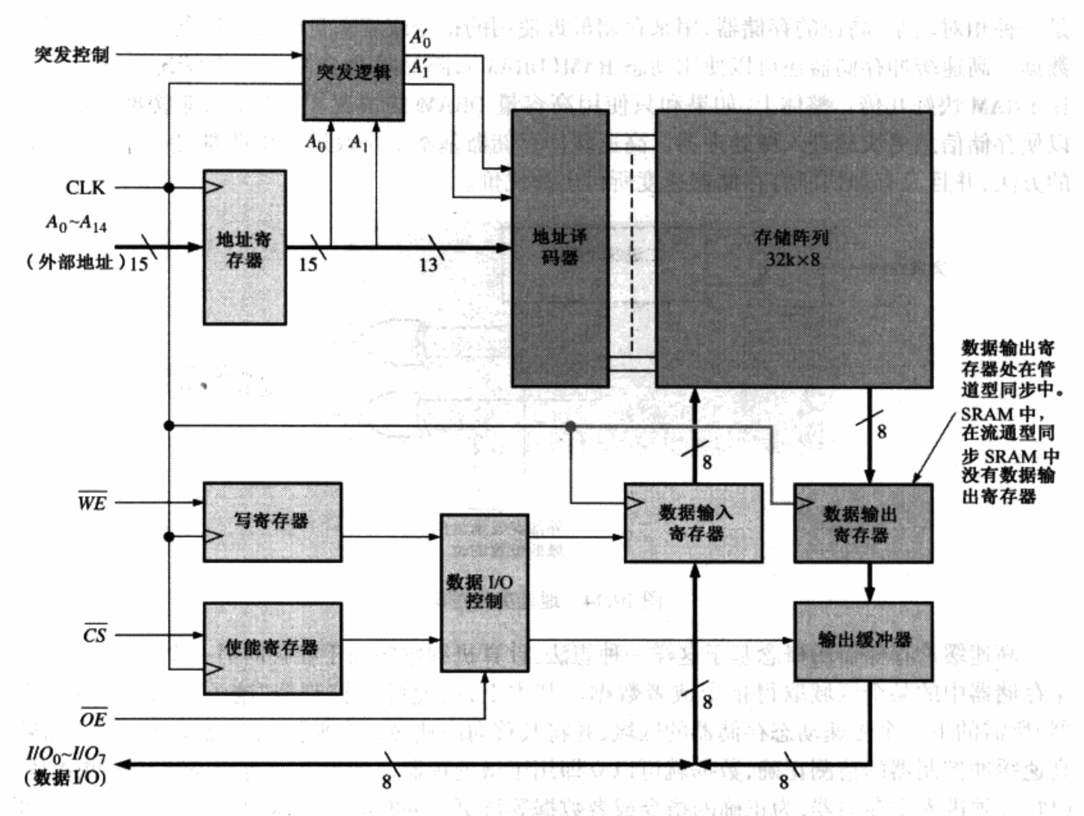


图 10.13 具有突发特性的同步 SRAM 的基本框图

同步 SRAM 的两种基本类型是流通型和管道型。流通型同步 SRAM 不具有数据输出寄存器，所以输出数据经过输出缓冲器异步流向数据 I/O 线。管道型同步 SRAM 具有数据输出寄存器，如图 10.13 所示，所以输出数据被同步置于数据 I/O 线上。

突发特性 如图 10.13 所示，同步 SRAM 一般具有地址突发特性，允许存储器使用单个地址在 4 个位置进行读或者写。当外部地址锁存到地址寄存器时，两个最低位置的地址位 A_0 和 A_1 就加在突发逻辑上。在连续的时钟脉冲作用下，通过把 00、01、10 和 11 加到这两个最低顺序地址位上，可以产生 4 个内部地址的序列。此序列总是开始于基地址，也就是保留在地址寄存器上的外部地址。

典型的同步 SRAM 中的地址突发逻辑由一个二进制计数器和异或门组成，如图 10.14 所示。对于 2 位突发逻辑，基地址位 A_2 到 A_{14} 再加两个突发地址位 A'_1 和 A'_0 就形成了内部突发地址序列。

为了开始这个突发序列，计数器处在它的 00 状态并且两个最低位置的地址位加在异或门的输入上。假设 A_0 和 A_1 都是 0，两个最低位置位表示的内部地址序列是 00、01、10 和 11。

10.2.5 高速缓冲存储器

SRAM 的一个主要应用是计算机中的高速缓冲存储器(简称高速缓存)。高速缓冲存储器

是一种相对较小、高速的存储器,用来存储最近使用的指令或者来自较大且较慢的主存储器的数据。高速缓冲存储器还可以使用动态 RAM(DRAM),随后将会介绍。一般情况下,SRAM 要比 DRAM 快好几倍。整体上,如果和只使用高容量 DRAM 的情况相比较,高速缓冲存储器可以使存储信息更快地进入微处理器。高速缓冲存储器基本上是改进系统性能的一个经济实惠的方法,并且没有加快所有存储器速度所付出的代价。

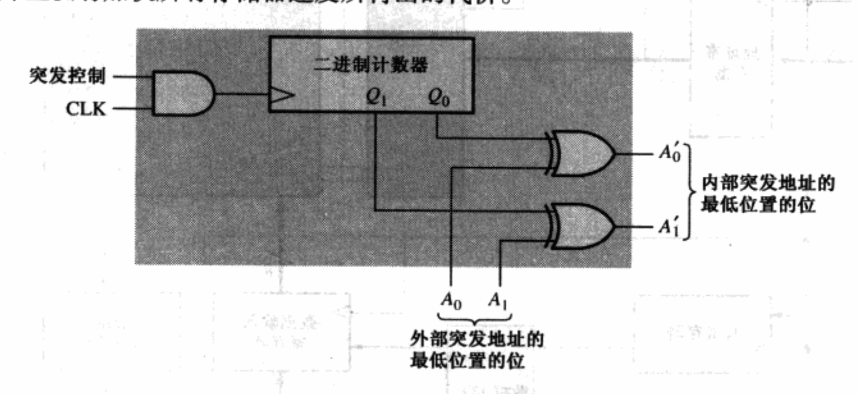


图 10.14 地址突发逻辑

高速缓冲存储器的概念基于这样一种想法:计算机程序倾向于在移向另一个区域之前,从主存储器中的某个区域取得指令或者数据。基本上,高速缓冲控制器"猜测"CPU(中央处理器)所需的下一个慢速动态存储器的区域,并将其移向高速缓冲存储器以备需要时使用。如果高速缓冲控制器的猜测正确,数据就可以立即用于微处理器。如果高速缓冲控制器猜测错误,CPU 必须进入主存储器,为正确的指令或者数据等待更长的时间。幸运的是,高速缓冲控制器在绝大多数情况下的"猜测"都是正确的。

高速缓存类比 描述高速缓冲存储可以使用许多类比,但是把它和家用电冰箱做比较可能是最有效的。可以认为家用电冰箱是某些食品项目的"高速缓存",而超市就是主存储器,它是保存所有食品的地方。每次想吃或者喝什么东西时,就可以先看看冰箱(高速缓存)里有没有想要的东西。如果有,便节约了许多时间。如果没有,就不得不花费额外的时间去超市(主存储器)购买。

L1 和 L2 高速缓存 第一级高速缓存(L1 高速缓存)常常集成到处理器芯片上,并且具有非常有限的存储容量。L1 高速缓存被称为主超高速缓存。第二级超高速缓存(L2 高速缓存)是一个独立的外存储器芯片或者处理器之外的芯片集合,并且通常具有比 L1 高速缓存更大的存储容量。L2 超高速缓存也被称为次级高速缓存。某些系统可能具有更高级的高速缓存(L3、L4 等),但是 L1 和 L2 是最常见的。同样,一些系统使用磁盘高速缓存来增强硬盘的性能,因为虽然 DRAM 要比 SRAM 慢得多,但是也要比硬盘驱动器快得多。图 10.15 解释了计算机系统中的 L1 和 L2 高速缓存。

10.2.6 动态 RAM(DRAM)存储单元

动态存储单元在一个很小的电容里而不是锁存器里存储一个数据位。这种类型单元的优点是,它非常简单,因此允许很大的内存阵列构建在一个芯片上,并且每位的成本很低。缺点

是存储电容不能长时间保持它的电荷,并且会丢失它所保存的数据位,除非对它的电荷进行定期刷新。为了刷新就需要附加的存储电路,这使得 DRAM 的操作复杂化。图 10.16 给出了一个典型的 DRAM 单元,由单个 MOS 晶体管(MOSFET)和一个电容组成。

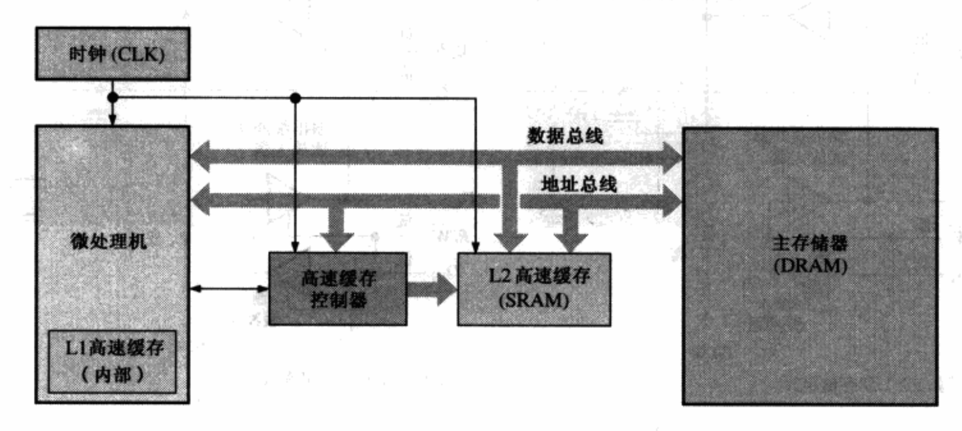


图 10.15　给出计算机系统中 L1 和 L2 高速缓存的框图

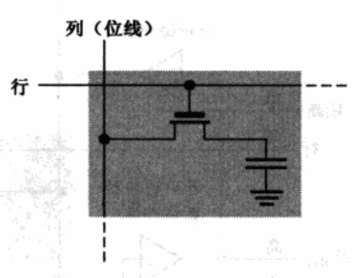


图 10.16　一个 MOS DRAM 单元

在这种类型的单元中,晶体管作为开关来使用。基本的简化操作如图 10.17 所示,并且如下所述。$R/\overline{W}$ 线(写模式)上的低电平使能三态输入缓冲器并且禁止输出缓冲器。对于即将写入单元中的一个 1 来说,D_{IN}(数据输入)线必须为高电平,必须由行线上的高电平使晶体管导通。晶体管相当于一个闭合开关,连接电容到位线上。这个连接允许电容充电到一个正电压,如图 10.17(a)所示。当 0 被存储时,低电平就加在 D_{IN}线。如果电容正在存储 0,它就保持非充电状态;如果它正在存储 1,电容就放电,如图 10.17(b)所示。当行线返回低电平时,晶体管截止,电容和位线断开,从而在电容中“俘获”电荷(1 或者 0)。

为了从单元中读出,$R/\overline{W}$(读/写)线就是高电平,使能输出缓冲器并且禁止输入缓冲器。当行线取高电平时,晶体管导通并将电容连接到位线上,因此也就连接到输出缓冲器(读放大器)上,所以数据位就出现在数据输出线(D_{OUT})上。这个过程如图 10.17(c)所示。

为了刷新存储单元,$R/\overline{W}$ 线是高电平,行线为高电平,而刷新线也是高电平。晶体管导通,把电容连接到位线上。输出缓冲器被开启,而存储的数据位加在刷新缓冲器的输入上,此缓冲器由刷新输入上的高电平开启。这就在相应存储位的位线上产生一个电压,因而重新对电容进行充放电。这个过程如图 10.17(d)所示。

10.2.7 基本 DRAM 的组成结构

DRAM 的主要应用是计算机的主存储器。DRAM 和 SRAM 之间的区别是存储单元的类型不同。正如我们所看到的那样,DRAM 的存储单元由一个晶体管和一个电容组成,这要比 SRAM 单元简单得多。这种组成结构允许 DRAM 中有更大的密度,因而在给定的芯片面积里有更大的位容量,但是存取时间慢了许多。

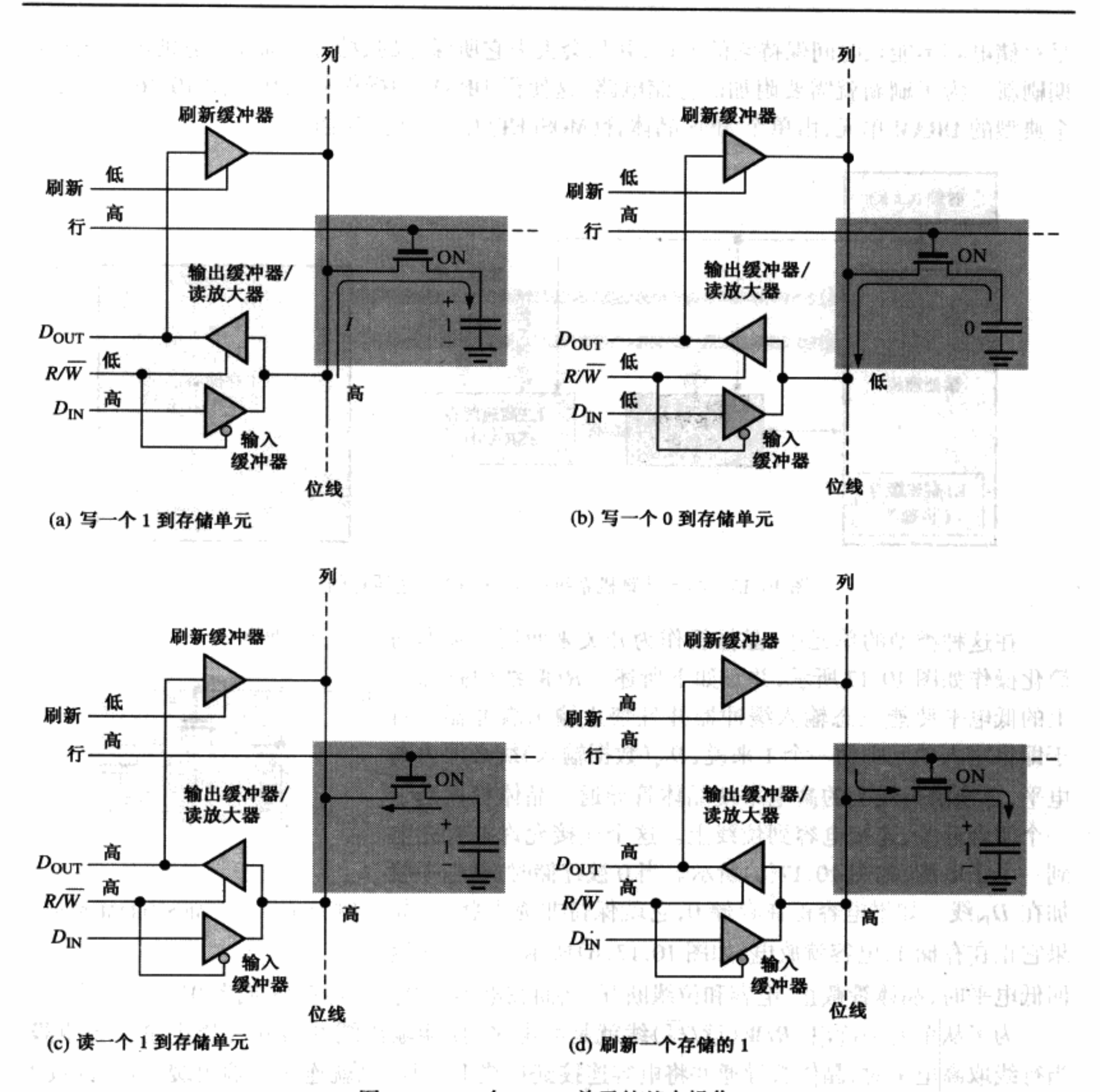


图 10.17 一个 DRAM 单元的基本操作

再一次,因为电容中存储的电荷会泄漏,DRAM 单元需要经常的刷新操作来保持存储的数据位。这个需求就导致了比 SRAM 中更加复杂的电路。我们使用一般的 1M×1 位 DRAM 作为例子,讨论大多数 DRAM 常见的几个特征。

地址多路复用 DRAM 使用一种称为地址多路复用的技术来减少地址线的数目。图 10.18给出了一个具有 1M×1 组成结构的 1 048 576 位(1 MB)DRAM 的框图。这里着重于浅色的方块,以解释地址多路复用。深色的方块表示刷新逻辑。

在存储周期的开始处,通过行地址选择($\overline{RAS}$)和列地址选择($\overline{CAS}$)而产生两个独立的 10 位地址字段,10 条地址线进行时分多路复用。首先,10 位行地址被锁存到行地址锁存器中。接下来,10 位列地址被锁存到列地址锁存器中。行地址和列地址被译码以后,在存储阵列中选择 1 048 576个地址(2^{20} = 1 048 576)中的一个地址。该地址多路复用操作的基本时序如图 10.19 所示。

图 10.18 1M × 1 DRAM 的简化框图

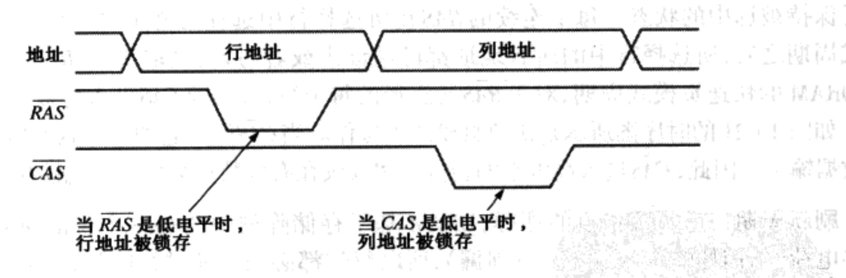


图 10.19 地址多路复用的基本时序

读和写周期 在每个读和写存储周期的开始，$\overline{RAS}$和$\overline{CAS}$都进入有效状态(低电平)以多路复用行和列地址到锁存器和译码器。对于读周期，$R/\overline{W}$ 输入为高电平。对于写周期，$R/\overline{W}$ 输入为低电平。这个过程如图 10.20 所示。

快速页模式 在先前所描述的一般读或者写周期中，特定存储位置的行地址首先由低电平有效$\overline{RAS}$载入，随后此位置的列地址由低电平有效$\overline{CAS}$载入。下一个位置由另外的$\overline{RAS}$和$\overline{CAS}$来选择，以此类推。

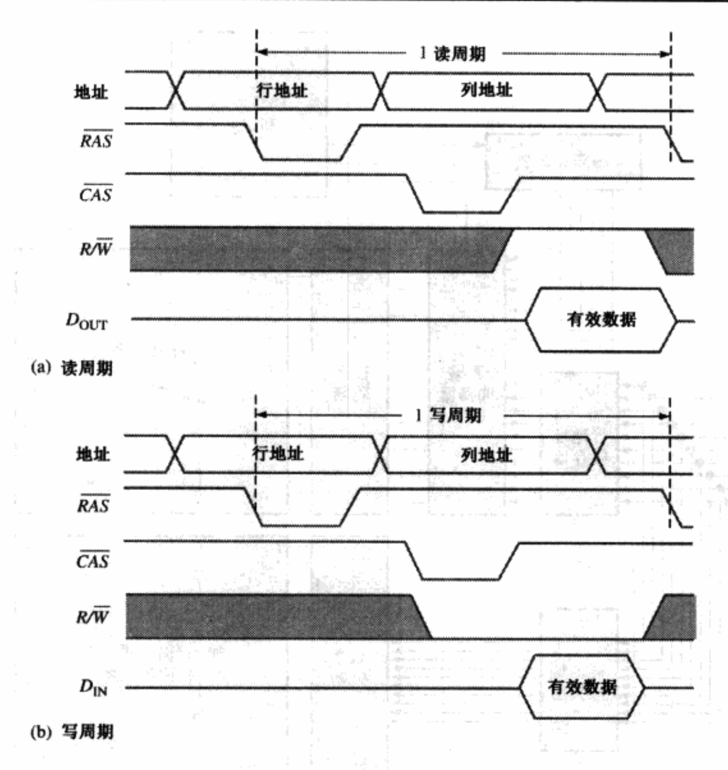


图 10.20 一般的读写周期的时序

"页"就是在单个行地址中可用的存储部分,由一行中的所有列组成。快速页模式允许在所选择行中的每个列地址上进行快速连续的读或者写操作。通过$\overline{RAS}$变为低电平并且在$\overline{CAS}$于高低电平之间切换期间保持低电平,行地址首先被载入。单个行地址被选择,并且在$\overline{RAS}$有效时保持被选中的状态。每个连续的$\overline{RAS}$在所选择行中选择另外的列。所以,在一个快速页模式周期之后,所选择行中的所有地址都已经读出或者写入,这取决于 $R/\overline{W}$。例如,图 10.18 中 DRAM 的快速页模式周期,对于$\overline{RAS}$所选择的每个行,都需要$\overline{CAS}$进入有效状态 1024 次。

如图 10.21 的时序图所示是快速页模式读操作。当$\overline{CAS}$进入它的无效状态(高电平)时,就禁止数据输出。因此,$\overline{CAS}$进入高电平的转换必须仅仅在有效数据被外部系统锁存以后才能发生。

刷新周期 正如所知道的那样,DRAM 基于存储阵列中的每一个位的电容电荷的存储。这些电荷随着时间和温度而减少(泄漏),所以每位都必须定期进行刷新(再充电)以保持正确的位状态。一般情况下,DRAM 必须每隔 8 ms 到 16 ms 刷新一次,虽然对于某些芯片来说,刷新周期可以超过 100 ms。

读操作自动刷新所选择行中的所有地址。但是,在典型的应用中,不能总是预期多久才会有一个读周期,因此不能依赖读周期足够频繁地出现来防止数据丢失。所以,必须在 DRAM 系统中实现特殊的刷新周期。

突发刷新和分布刷新是刷新操作的两种基本刷新模式。在突发刷新中,存储阵列中的所有行在每个刷新周期都会被连续刷新。对于刷新周期为 8 ms 的存储器,所有行的突发刷新每 8 ms 发生一次。在突发刷新周期内,一般的读和写操作都会被暂停。在分布刷新中,每一行

都在正常的读或者写周期之间散布的时间间隔内被刷新。例如，图 10.18中的存储器具有 1024 行。作为一个例子，对于 8 ms 的刷新周期，当使用分布刷新时，每一行都必须每隔 8 ms/1024 = 7.8 μs 刷新一次。

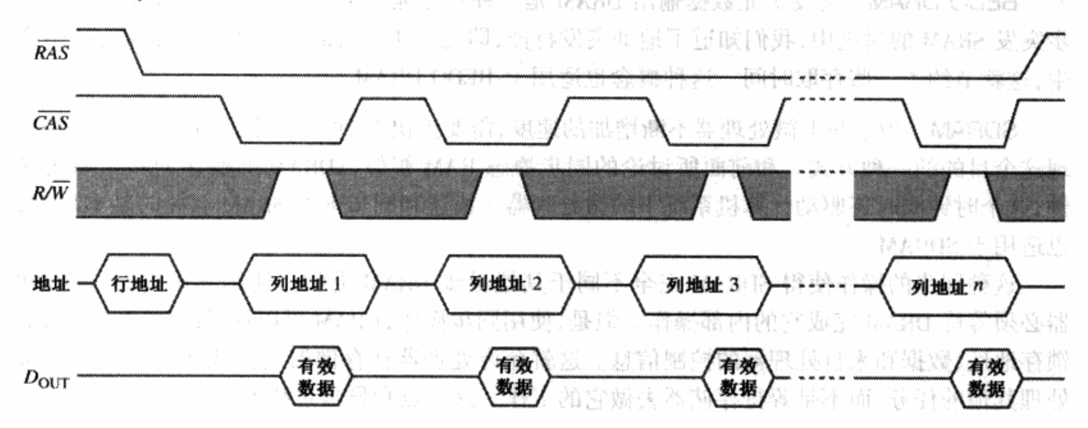


图 10.21 快速页模式读操作的时序

刷新操作的两种类型是：仅$\overline{RAS}$刷新和$\overline{CAS}$先于刷新。仅$\overline{RAS}$刷新由$\overline{RAS}$转换到低电平(有效)状态组成，其锁存将要刷新的行地址，而$\overline{CAS}$在整个周期内保持为高电平(无效)。使用外部计数器提供这种操作类型的行地址。

在$\overline{RAS}$刷新前，通过$\overline{CAS}$变为低电平(在$\overline{RAS}$变为低电平前)，使得$\overline{CAS}$初始化。这个序列使得内部刷新计数器工作，产生即将刷新的行地址。这个地址由数据选择器转换到行译码器中。

10.2.8 DRAM 的类型

现在，我们已经学习了 DRAM 的基本概念，下面简要地看一下主要的类型，它们是快速页模式(FPM)DRAM、扩充数据输出(EDO)DRAM、突发扩充数据输出(BEDO)DRAM 和同步(S)DRAM。

FPM DRAM 先前描述了快速页模式的操作。这种类型的 DRAM 是最常见的，并且一直应用于计算机中，直至开发出了 EDO DRAM。回顾一下，存储器中的一页是指一个行地址内所包含的所有列地址。

FPM DRAM 的基本概念基于这样一种可能性：接下来要访问的几个内存地址位于同一行中(同一页中)。幸运的是，发生这样的情况所占的比例很高。FPM 比纯粹的随机存取要节省时间，因为在 FPM 中，行地址只被指定一次，就可以访问几个连续的列地址；而对于纯粹的随机访问来说，对于每个列地址都要指定一个行地址。

记得在快速页模式读操作中，$\overline{CAS}$信号在可以进入它的无效状态之前不得不等待，直到来自给定地址的有效数据被外部系统(CPU)接收(锁存)为止。当$\overline{CAS}$进入它的无效状态后，数据输出就被禁止。这就意味着下一个列地址不能发生，直到来自当前列地址中的数据被传送到 CPU 之后为止。这就限制了页内的列寻址速度。

EDO DRAM 扩充数据输出 DRAM，有时也称为超页模式 DRAM，非常类似于 FPM DRAM。关键的区别是 EDO DRAM 中的$\overline{CAS}$信号进入它的无效状态时，并不禁止输出数据，因为来自当

前地址的有效数据可以保存下来，直至$\overline{CAS}$再次有效。这就意味着在外部系统接收当前有效数据之前，可以访问下一个列地址。这种方式用以加速存取时间。

BEDO DRAM 突发扩充数据输出 DRAM 是一种具有地址突发能力的 EDO DRAM。从同步突发 SRAM 的讨论中，我们知道了地址突发特征，即允许 4 个内部地址从单个外部地址中产生，这就节约了一些存取时间。这种概念也适用于 BEDO DRAM。

SDRAM 为了跟上微处理器不断增加的速度，需要更快的 DRAM。同步 DRARAM 就是实现这个目的的一种方式。和前面所讨论的同步静态 RAM 相似，SDRAM 的操作同步于系统时钟，这个时钟同时还驱动计算机系统中的微处理器。描述和同步突发 SRAM 有关的基本思想，也适用于 SDRAM。

这种同步的操作使得 SDRAM 完全不同于其他异步 DRAM 类型。使用异步存储，微处理器必须等待 DRAM 完成它的内部操作。但是，使用同步操作，DRAM 可以在系统时钟的控制下锁存地址、数据和来自处理器的控制信息。这就允许处理器在存储器的读或者写操作进行时处理其他的任务，而不是等待存储器去做它的工作。这一点和异步系统不同。

10.3 只读存储器(ROM)

ROM 可以永久或者半永久地保存数据，这些数据可以从内存中读出，但是不能改变或者没有专门设备就不能改变。ROM 存储的那些数据在系统应用中被重复地使用，比如表格，变换或者用以系统初始化和操作的编程指令。当电源关闭时，ROM 仍然保存着存储的数据，因而是非易失性存储器。

学完本节以后，应当能够

- 列出 ROM 的类型
- 描述基本的掩模 ROM 存储单元
- 解释怎样从 ROM 中读出数据
- 讨论典型 ROM 的内部组成结构
- 讨论一些 ROM 的应用

10.3.1 ROM 系列

图 10.22 给出了半导体 ROM 是怎样归类的。掩模 ROM 在生产过程中将数据永久地存储在存储器中。PROM 也就是可编程 ROM，用户在专业化设备的帮助下，对数据进行电存储。掩模 ROM 和 PROM 可以是 MOS 或者双极性技术。EPROM 也就是可擦除 PROM，是严格的 MOS 设备。UV EPROM 是用户电可编程的，但是存储的数据必须暴露在紫外灯下几分钟的时间，才可以被擦除掉。电可擦 PROM(EEPROM 或者 E^2PROM)可以在几毫秒内被擦除掉。

10.3.2 掩模 ROM

掩模 ROM 通常简单地指 ROM。它在生产过程中被永久编程，以提供广泛使用的标准函数，比如通用变换，或者提供用户指定的函数。一旦存储器被编程之后，它就不能改变了。大多数 IC ROM 使用行/列连接处晶体管连接的存在或者缺失来表示 1 或者 0。

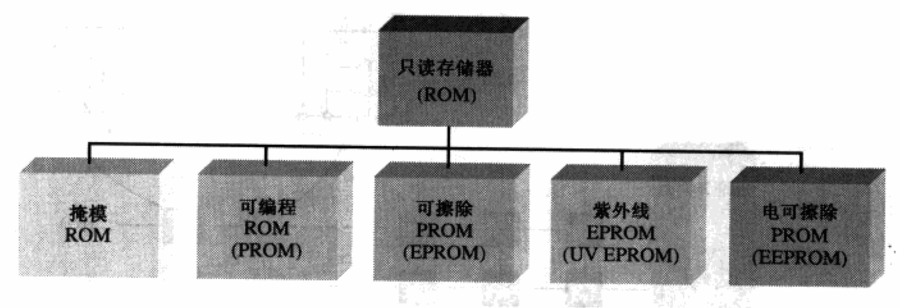


图 10.22　ROM 系列

图 10.23 给出了 MOS ROM 单元。行线到晶体管栅极连接的存在表示该位置的一个 1,因为当行线为高电平时,所有栅极与这条行线连接的晶体管都导通,并将高电平(1)连接到相关的列线上。在没有栅极连接的行/列连接处,当对行进行寻址时,列线保持为低电平(0)。

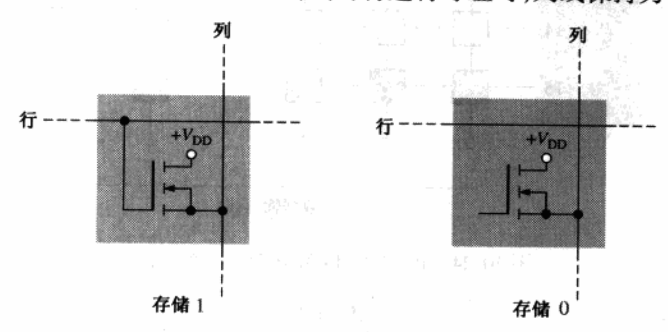


图 10.23　ROM 单元

10.3.3　简单的 ROM

为了解释 ROM 的概念,图 10.24 给出了一个简化的 ROM 阵列。深色方格表示存储的是 1,而浅色方格表示存储的是 0。基本读操作如下所示:当二进制地址码加到地址输入线时,相应的行线就变为高电平。这个高电平在每个结点(单元)处通过晶体管连接到列线上,该单元存储的是 1。在每个存储 0 的单元处,由于终端电阻,列线保持为低电压。这些列线构成了数据输出。存储在所选行中的 8 个数据位出现在数据输出线上。

正如所看到的那样,图 10.24 中的 ROM 被组成为 16 个地址,每个地址保存 8 个数据位。因此,它是 16×8(16 乘以 8)ROM,并且它的总容量是 128 位或者 16 字节。ROM 可以用做查找表格(LUT),用于代码变换和逻辑函数的产生。

例 10.1　给出一个基本 ROM,和图 10.24 中的 ROM 相似,为 4 位二进制到格雷码的转换编程。

解:复习第 2 章的格雷码。开发出来的表 10.1 用在编程 ROM 中。

结果中的 16×4 ROM 如图 10.25 所示。可以看到地址输入线的二进制码产生输出线(列)上相应的格雷码。例如,当二进制数 0110 加在地址输入线上时,存储格雷码 0101 的地址 6 被选中。

相关问题:利用图 10.25,当二进制码 1011 加到地址输入线上时,确定格雷码的输出。

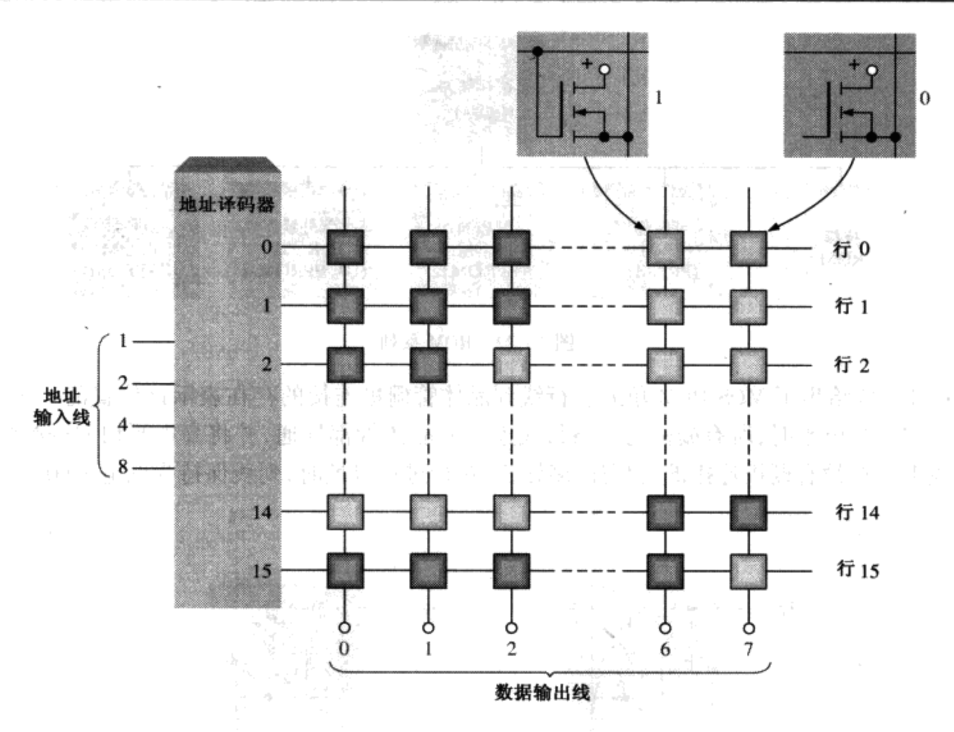


图 10.24 16×8 位 ROM 阵列的一个表示

表 10.1

二进制				格雷码			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

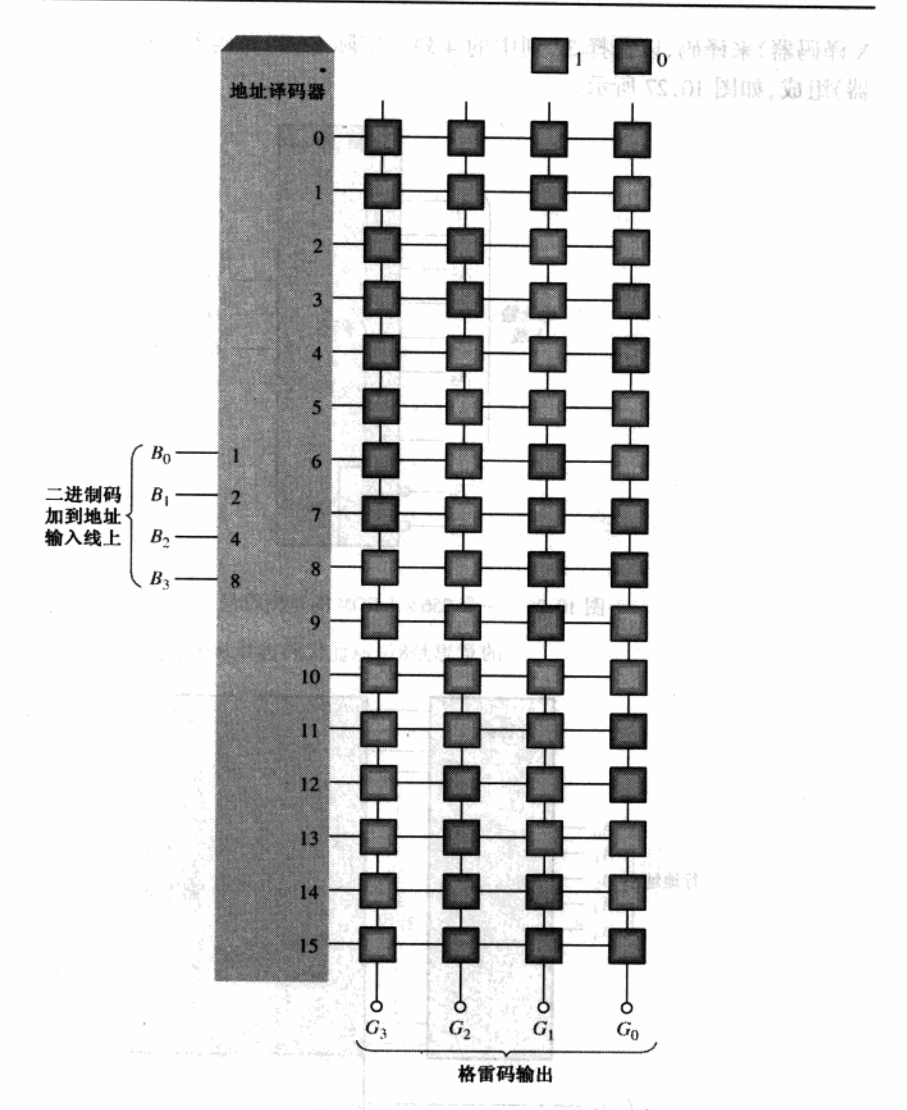


图 10.25　作为二进制到格雷码转换器的一个 ROM 编程的表示

10.3.4　内部 ROM 组成结构

大多数 IC ROM 要比刚刚呈现的基本简化例子具有更加复杂的内部组成结构。为了解释 IC ROM 是怎样构建的，使用一个具有 256×4 组成结构的 1024 位芯片。逻辑符号如图 10.26 所示。当 256 个二进制代码(8 位)中的任何一个加到地址线上时，如果芯片使能输入为低电平，那么 4 个数据位就会出现在输出上。(因为 256 个地址需要 8 条地址线。)

虽然芯片的 256×4 组成结构说明在存储阵列中具有 256 行和 4 列，但实际上并不是这样的。存储单元阵列实际上是一个 32×32 的矩阵(32 行和 32 列)，如图 10.27 中的框图所示。

图 10.27 中的 ROM 的工作如下：8 条地址线中的 5 条(A_0 到 A_4)由列译码器(经常称为 Y 译码器)来译码，以选择 32 列中的 1 列。8 条地址线中的 3 条(A_5 到 A_7)由列译码器(经常称为

X 译码器)来译码，以选择 32 列中的 4 列。实际上，列译码器由四个 8 选 1 译码器(数据选择器)组成，如图 10.27 所示。

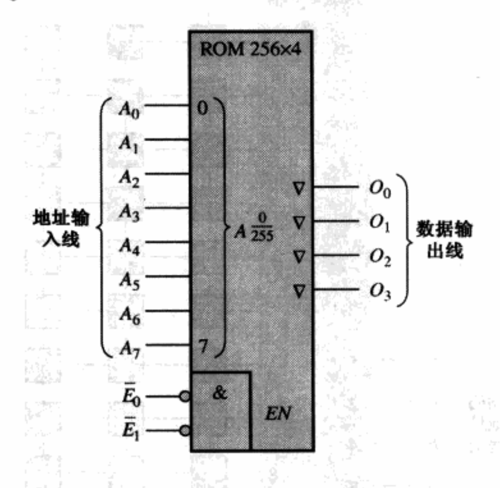


图 10.26　一个 256×4 ROM 的逻辑符号。$A\frac{0}{255}$指示符的意思是8位地址代码选择从0到255的地址

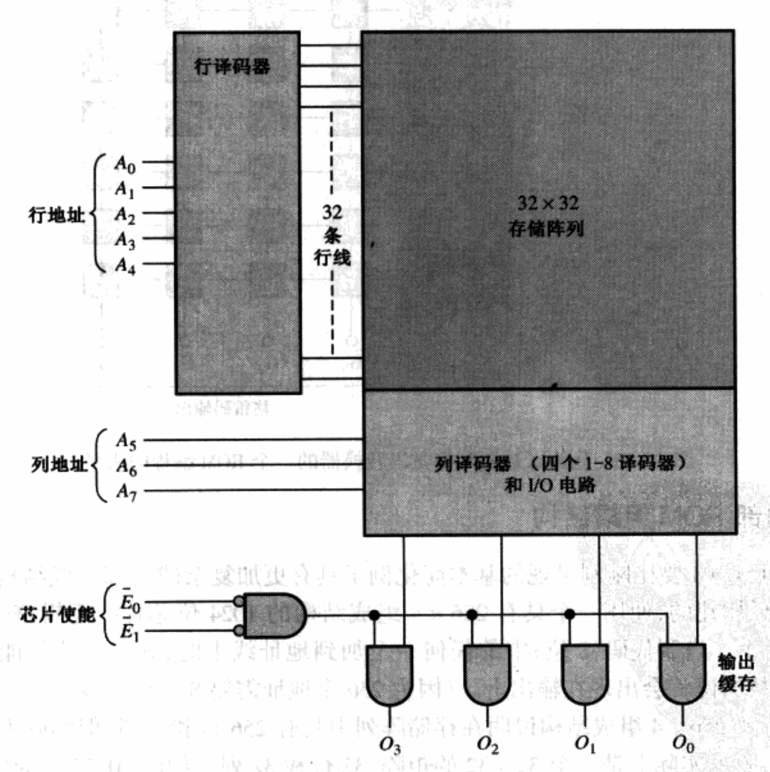


图 10.27　基于 32×32 阵列的具有 256×4 组成结构的 1024 位 ROM

这个结构的应用结果是，当使用一个 8 位地址代码（A_0 到 A_7）时，一个 4 位数据字就出现在数据输出上，这时芯片使能线（$\bar{E}_0$ 和 $\bar{E}_1$）为低电平，以启动输出缓冲器。这种类型的内部组成结沟（体系结构）是典型的不同容量的 IC ROM。

10.3.5 ROM 存取时间

解释 ROM 存取时间的典型时序图如图 10.28 所示。ROM 的存取时间是 t_a，即从输入线上有效地址码的应用到有效输出数据的出现所需的时间。当有效地址已经处在输入线上时，存取时间的测量也可以按照从芯片使能（$\bar{E}$）输入到有效输出数据的出现这段时间。

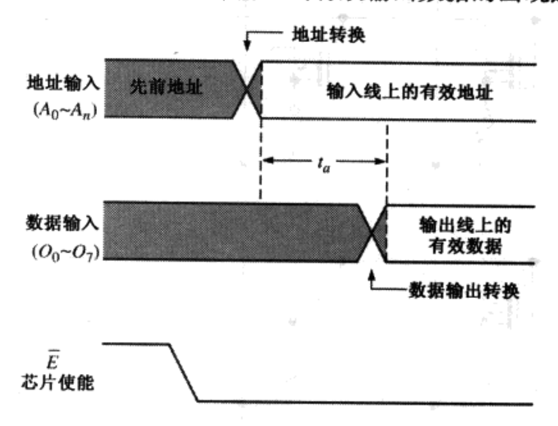


图 10.28 芯片使能有效，从地址改变到数据输出的 ROM 存取时间（t_a）



计算机小知识

在个人计算机中，ROM 被用来存储 BIOS(基本输入/输出系统)。BIOS 是一些程序，用以为计算机执行基本的监管和支持功能。例如，存储在 ROM 中的 BIOS 程序控制某种电视监视器的功能、提供磁盘格式化、为输入扫描键盘和控制某种打印机的功能。

10.4 可编程 ROM(PROM 和 EPROM)

可编程 ROM(PROM)一旦被编程之后，就基本上和掩模 ROM 相同。正如前面所学到的，ROM 是一种可编程的逻辑芯片，PROM 与其码区别是在生产商那里未编程，用户能够对其进行编程以满足用户方面的需求。

学完本节以后，应当能够

- 区分掩模 ROM 和 PROM
- 描述基本的 PROM 存储单元
- 讨论 EPROM，包括 UV EPROM 和 EEPROM
- 分析 EPROM 编程周期

10.4.1 PROM

PROM 使用一种熔丝过程的方法来存储位，以存储器的连接被烧断或者保持完好来表示 0 或者 1。熔丝过程是不可逆的；一旦 PROM 被编程之后，就不能改变了。

图 10.29 解释了一个具有熔丝连接的 MOS PROM。这个熔丝连接被安装到 PROM 中,位于每个单元的晶体管的源极和它的列线之间。在编程过程中,一个足够大的电流注入,通过熔丝连接将其烧断,从而创建一个存储的 0。如果连接保持完好,就是存储 1。

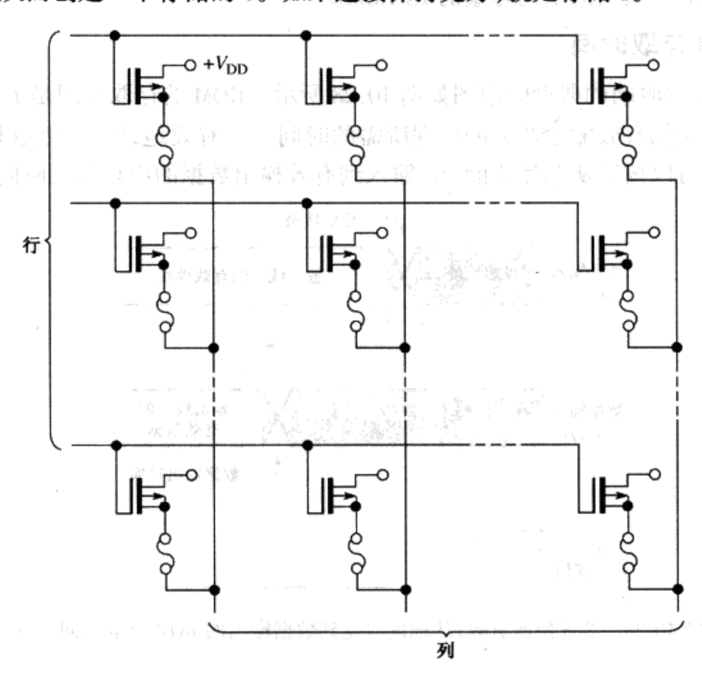


图 10.29 具有熔丝连接的 MOS PROM(所有的漏极共同连接到 V_{DD})

用在 PROM 中的三个基本熔丝技术是金属连接、硅连接和 *pn* 结。每种技术的简短描述如下:

1. 金属连接由诸如镍铬合金之类的材料构成。存储阵列中的每个位都由独立的连接表示。在编程期间,连接要么是“吹开的”断开要么保持完好。首先对给定的单元寻址,然后迫使足够量的电流经过连接而造成它断开。
2. 硅连接由狭窄的、有凹槽的多晶硅条带组成。这些熔丝的编程需要将足够量的电流通过它们,才能熔断连接。这些电流在熔丝部位产生高温,从而使硅氧化并在刚才断开的连接周围形成绝缘层。
3. 短路结或者雪崩诱导迁移技术,基本上由两个背靠背排列的 *pn* 结组成。在编程期间,二极管的一个结崩塌,结果电压和热量引起铝离子迁移和结的短路。其余的结随后被用做正偏压二极管,以表示一个数据位。

10.4.2 EPROM

EPROM 是可擦除 PROM。和普通的 PROM 不同,如果存储阵列中现有的程序首先被擦除,EPROM 可以被重编程。

EPROM 使用具有绝缘栅极结构的 NMOSFET。绝缘晶体管的栅极没有电的连接,从而可以在无限长的时间里存储一个电子电荷。这种类型的阵列中的数据位由栅极是否存储电荷来表示。数据位的擦除是移走栅极电荷的过程。

可擦除 PROM 的两种基本类型是紫外线可擦除 PROM(UV EPROM)和电可擦 PROM(EEPROM)。

UV EPROM 可以通过封装上的透明石英盖来识别 UV EPROM 芯片,如图 10.30 所示。紫外线 EPROM 的场效应管中的绝缘栅极“浮动”在氧化绝缘材料内。编程过程使得电子从浮动栅极中移走。擦除是通过封装顶上的石英窗口,把存储阵列芯片暴露在高密度紫外辐射下而完成的。存储在栅极上的正电荷在几分钟到一小时的暴露时间后得到中和。典型的 UV EPROM 由图 10.31 中的逻辑图表示。其操作是其他不同大小的典型 UV EPROM 的代表。正如逻辑符号所给出的那样,这个芯片具有 2048 个地址($2^{11}=2048$),每一个具有 8 个位。注意 8 个输出是三态(▽)的。

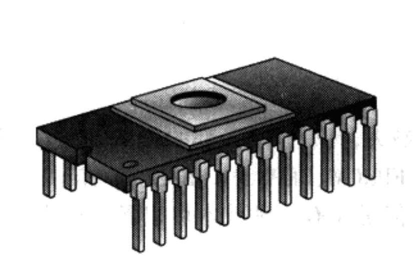


图 10.30 紫外线可擦除 PROM 的封装

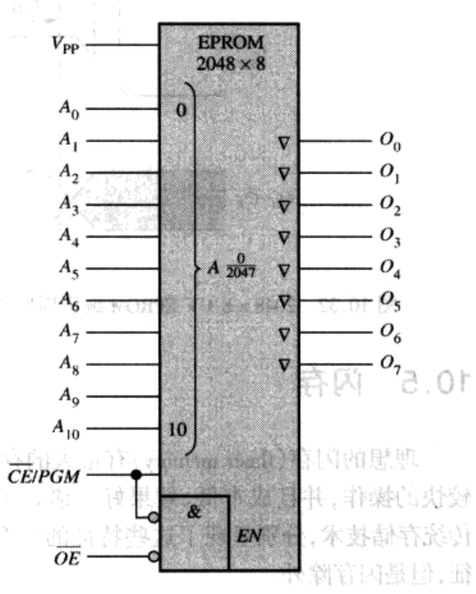


图 10.31 2048 × 8 UV EPROM 的逻辑符号

为了从存储器中读出数据,输出使能输入($\overline{OE}$)必须为低电平,而断电/编程($\overline{CE}/PGM$)输入为低电平。为了擦除存储的数据,芯片通过透明盖暴露于高密度紫外线下。典型的紫外线灯将会在 20 ~ 25 分钟内把数据擦除掉。和大多数的 UV EPROM 一样,擦除后所有的位都是 1。正常的环境光线包含真实的紫外波长的光,可以在一段时间内完成擦除。因此,封装上的透明盖必须保持掩盖。

为了对芯片进行编程,将一个直流高压加到 V_{PP}上,并且$\overline{OE}$为高电平。即将编入给定地址的 8 个数据位加在输出(Q_0 到 Q_7)上,而地址在输入 A_0 到 A_{10}上进行选择。接下来,一个高电平脉冲加到输入$\overline{CE}/PGM$。地址可以在任意的顺序下进行编程。

这个编程过程的时序图如图 10.32 所示。这些信号一般由 EPROM 编程器产生。

EEPROM 电可擦除 PROM 可以利用电脉冲进行擦除和编程。由于它可以电写入和电擦除,所以 EEPROM 可以在重编程的电路中被快速地编程和擦除。

EEPROM 的两种类型是浮栅 MOS 和金属氮化物 - 氧化物硅(MNOS)。浮栅结构中控制栅极上施加的电压允许在浮栅中存储或者移走电荷。

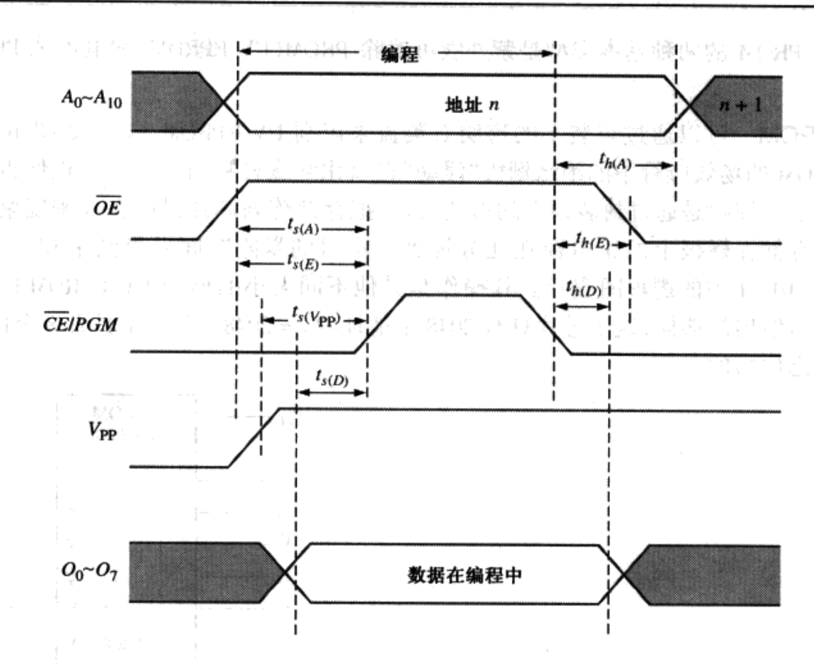


图 10.32 2048 × 8 UV EPROM 编程周期的时序图,指出了关键的建立时间(t_s)和保持时间(t_h)

10.5 闪存

理想的闪存(flash memory)有很大的存储容量,非易失,具有系统内部的读和写能力及相对较快的操作,并且成本低、效果好。诸如 ROM、PROM、EPROM、EEPROM、SRAM 和 DRAM 这样的传统存储技术,分别呈现了这些特征的一个或者多个,但是没有一种技术可以具有所有这些特征,但是闪存除外。

学完本节以后,应当能够

- 讨论闪存的基本特征
- 描述闪存单元的基本操作
- 闪存和其他类型的存储器的比较

闪存是高密度的非易失性读/写存储器(高密度意味着较大的位存储容量),也就是说,数据可以在没有电源的情况下永远地存储下来。它们有时用来取代便携式计算机的软盘或者小容量硬盘。

高密度的意思是芯片上给定的表面区域内可以容纳大量的单元;也就是说,密度越高,给定尺寸的芯片所能存储的位也就越多。在闪存中使用由单浮栅 MOS 晶体管组成的存储单元,这样高密度得以实现。数据位的存储依据存储 0 还是 1,对应于浮栅上有电荷或者没有电荷。

10.5.1 闪存单元

闪存中的单个晶体管单元如图 10.33 所示。层叠的栅 MOS 晶体管由一个控制栅极和一个浮栅及漏极和源极组成。一个足够的电压加在控制栅极上,导致浮栅存储电子(电荷)。当

浮栅上有更多电荷时，就存储 0，而当电荷很少或者没有电荷时，就存储 1。当读操作期间加上控制电压，这时出现在浮栅上的电荷的多少取决于晶体管是否导通和从漏极到源极的导通电流。

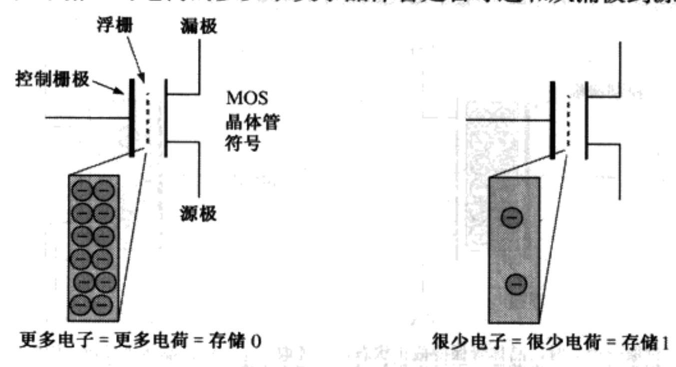


图 10.33 闪存中的存储单元

10.5.2 基本闪存操作

在闪存中有三种主要的操作：编程操作、读操作和擦除操作。

编程 初始时，所有的单元都处在 1 状态，因为在先前的擦除操作中把每个单元的电荷移走了。编程操作在要存储 0 的那些单元的浮栅中加入电子（电荷），那些存储 1 的单元中则不加入电荷。在编程期间，施加一个相对于源极的足够的正电压于控制栅极，以此把电荷吸引到浮栅上，如图 10.34 所示。一旦编程之后，在没有外部电源的情况下，单元上的电荷可以保留 100 年。

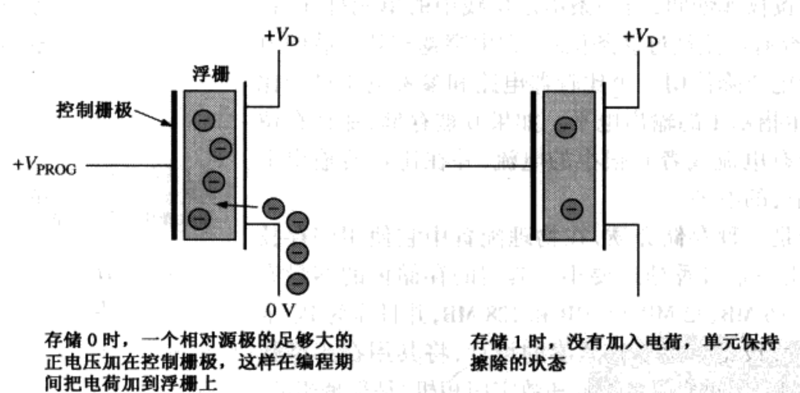


图 10.34 编程操作期间闪存单元存储 0 或者 1 的简单解释

读 在读操作期间，正电压加在控制栅极上。单元浮栅所呈现的电荷量取决于加在控制栅极的电压是否使晶体管导通。如果存储 1，控制栅极的电压就足以使晶体管导通。如果存储 0，晶体管将不会导通，因为控制栅极的电压不足以克服存储在浮栅中的负电荷。把浮栅上的电荷考虑为一个电压源，对抗读操作期间加在控制栅极上的电压。所以和存储 0 相对应的浮栅电荷，会阻止控制栅电压达到能使晶体管导通的电压阈值，而和存储 1 相对应的较少或者零电荷则允许控制栅极的电压超过使晶体管导通的电压阈值。

当晶体管导通时,就会有电流从漏极流向单元晶体管的源极。这个电流的出现就表示一个 1,没有电流就表示一个 0。这个基本思想如图 10.35 所示。

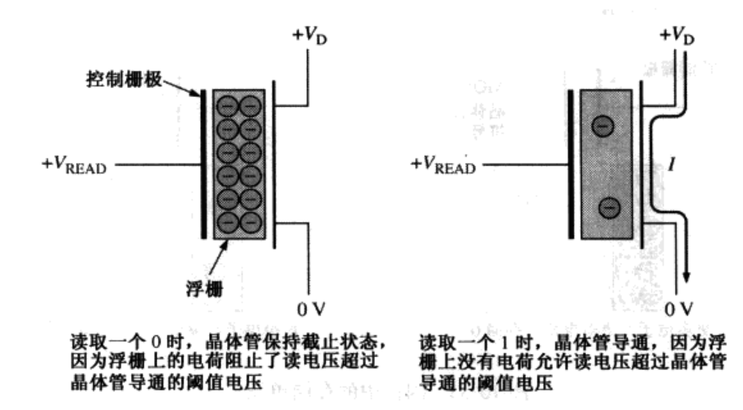


图 10.35 一个阵列中闪存单元的读操作

擦除 在擦除操作期间,电荷被从所有的存储器单元中移走。一个相对于控制栅极的足够大的正电压加在晶体管源极。这和在编程中使用的极性是相反的。此电压从浮栅中吸引电子,并耗尽它的电荷,如图 10.36 所示。闪存总是在重编程之前被擦除。

10.5.3 基本闪存阵列

闪存单元的简化阵列如图 10.37 所示。每次只访问一条行线。如读操作期间,当一条给定位线中的单元打开(存储 1)时,就会有电流经过这条位线,产生穿越有效负载的电压降。这个电压降使用一个比较器电路和参考电压进行比较,并且产生指示 1 的输出电平。如果 0 被存储,那么在位线中就不会有电流或者有很小的电流,并在比较器输出上产生一个相反的电平。

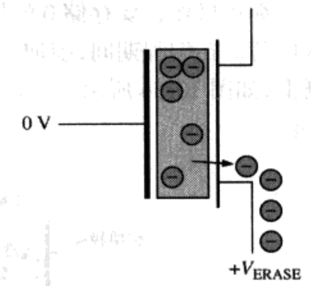


图 10.36 擦除期间从单元中移走电荷的简单解释

存储棒是一种存储介质,在物理配置中它使用闪存技术,其尺寸比一条口香糖还要小。典型的存储棒的容量有 4 MB、8 MB、16 MB、32 MB、64 MB 和 128 MB,并且作为 PC 卡适配器的配套设备。由于它紧凑的设计,将其用在小型数字电子产品中(例如笔记本电脑和数字照相机)是很理想的。

10.5.4 闪存和其他存储器的比较

把闪存和所熟悉的其他类型的存储器进行比较。

闪存与 ROM、EPROM 和 EEPROM 只读存储器是高密度,非易失的芯片。但是,一旦编程之后,ROM 的内容就不能改变。同样,初始编程也是费时费钱的过程。

虽然 EPROM 是高密度、非易失的存储器,但是它只能通过将存储内容移出系统和使用紫外线灯来擦除。EPROM 只能使用专门的设备来重编程。

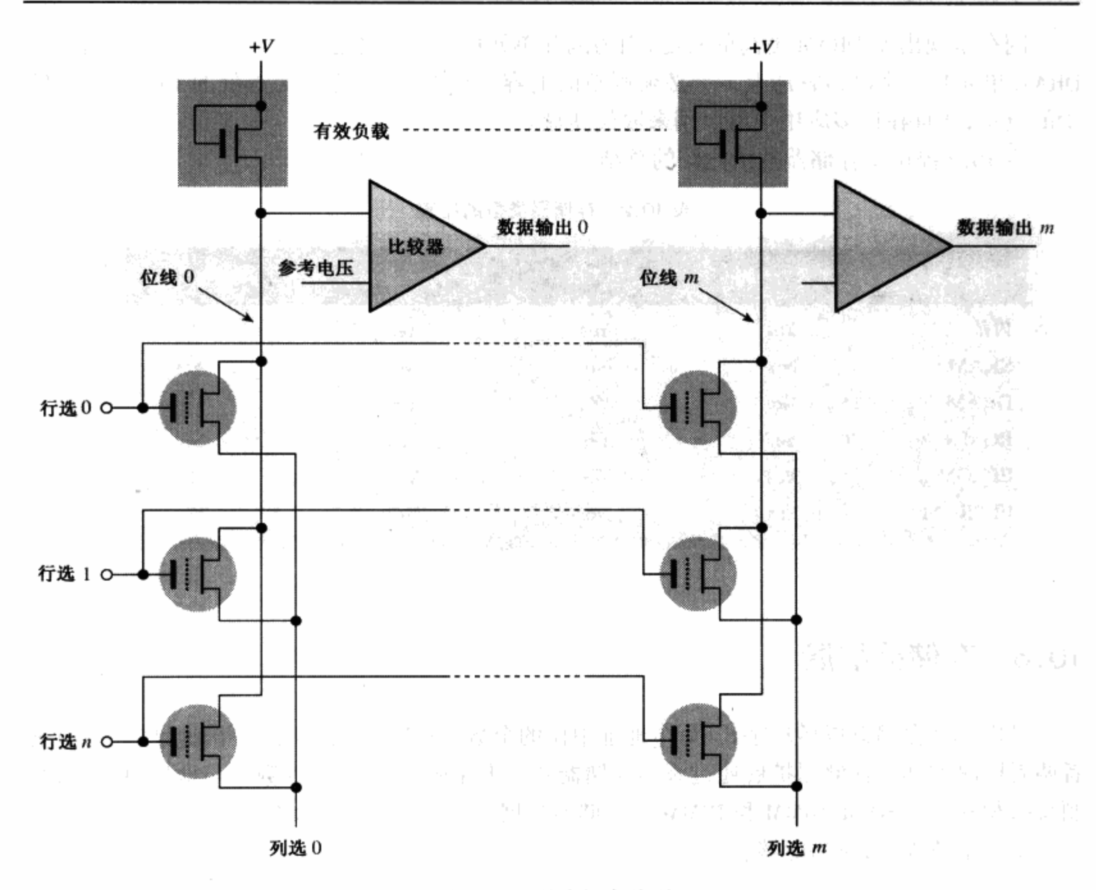


图 10.37　基本闪存阵列

EEPROM 比 ROM 或者 EPROM 具有更加复杂的单元结构，并且它的密度并不是太高，虽然可以在存储内容不移出系统的情况下进行重编程。由于它密度低，每位的成本要比 ROM 或者 EPROM 高许多。

闪存可以很容易地在系统中进行重编程，因为它本质上是一种读/写芯片。闪存的密度可以与 ROM 和 EPROM 相媲美，因为它们都具有单个晶体管单元。闪存（和 ROM、EPROM 或者 EEPROM 一样）是非易失性的，允许在电压断开的情况下永久地存储数据。

闪存与 SRAM　正如所学到的那样，静态随机存储器是易失性的读/写芯片。SRAM 需要恒定的电源来保持它所存储的数据。在许多应用中，如果主电源被关闭，可以使用备用电池来防止数据丢失。但是，由于电池故障总是可能的，存储数据的永远保存是不能保证的。因为 SRAM 中的存储单元基本上是由几个晶体管组成的触发器，所以密度相对较低。

闪存也是读/写芯片，但是和 SRAM 不同，它是非易失性的。同样，闪存比 SRAM 具有更高的密度。

闪存与 DRAM　动态随机存储器是易失性高密度读/写芯片。DRAM 不仅需要恒定的电源来保持数据，还需要经常刷新所保存的数据。在许多应用中，对于 DRAM，必须使用诸如硬盘之类的备份存储。

闪存呈现出比 DRAM 更高的密度,因为闪存单元由一个晶体管组成,并且不需要刷新,而 DRAM 单元是一个晶体管加上一个必须刷新的电容。典型地,闪存要比等价的 DRAM 耗费较少的电量,并且在许多应用中可以用来取代硬盘。

表 10.2 提供了存储器技术比较的总结。

表 10.2 存储器类型的比较

存储器类型	非易失性的	高密度	一个晶体管单元	可以写入系统
闪存	Yes	Yes	Yes	Yes
SRAM	No	No	No	Yes
DRAM	No	Yes	Yes	Yes
ROM	Yes	Yes	Yes	No
EPROM	Yes	Yes	Yes	No
EEPROM	Yes	No	No	Yes

10.6 存储器扩展

可以扩展存储器以增加字长(每个地址中位的个数)或者增加字容量(不同地址的个数)或者两者同时增加。存储器扩展通过添加存储器芯片来完成,添加的芯片数目与地址、数据和控制总线相匹配。SIMM、DIMM 和 RIMM 是存储器扩展模块的类型,这里将会介绍。

学完本节以后,应当能够

- 定义字长扩展
- 给出怎样扩展存储器字长的方法
- 定义字容量扩展
- 展示怎样扩充存储器的字容量

10.6.1 字长扩展

为了增加存储器的字长,必须增加数据总线中位的个数。例如,8 位字长可以通过使用两个存储器来实现,每个存储器都具有 4 位的字,如图 10.38(a)所示。正如在图 10.38(b)看到的那样,16 位地址总线通常连接到两个存储器上,使得组合而成的存储器仍然具有和每一个存储器相同的地址数(2^{16} = 65 536)。来自两个存储器的 4 位数据总线组合在一起而形成一个 8 位数据总线。这样,当选中一个地址时,就会在数据总线上产生 8 个位——从每个存储器得到 4 位。例 10.2 给出了详细的 65 536 × 4 到 65 536 × 8 的扩展。

例 10.2 扩展图 10.39 中的 65 536 × 4 ROM(64k × 4)形成一个 64k × 8 ROM。注意“64k”是 65 536的缩写。为什么不是“65k”? 可能因为 64 也是 2 的幂。

解:两个 64k × 4 ROM 按照如图 10.40 所示的那样连接在一起。注意 ROM 1 和 ROM 2 中的一个特定地址同时得到访问。在 ROM 1 中,4 个位形成了一个选中的地址,4 个位来自

ROM 2 中的相应地址,这两个地址并行输出,在数据总线上形成一个 8 位字。同时还要注意芯片使能线上的低电平,其形成一个简单的控制总线 $\bar{E}$,同时使两个存储器工作。

相关问题:描述怎样扩充 64k × 1 ROM 到 64k × 8 ROM。

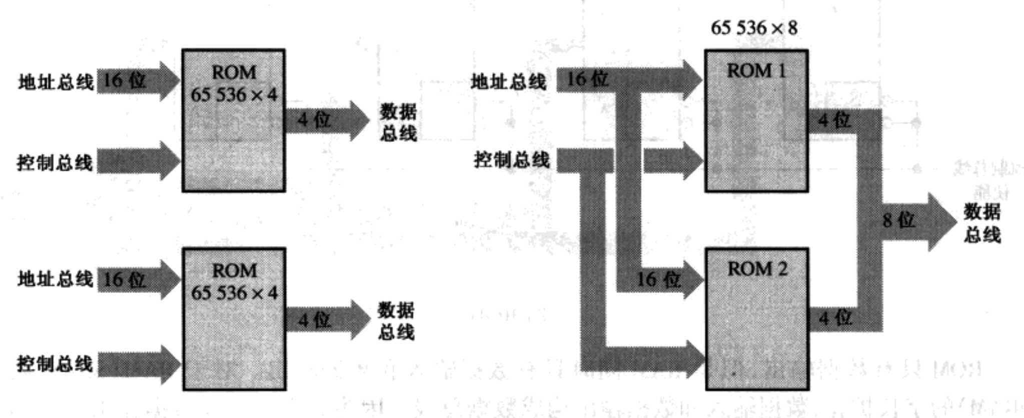


图 10.38　两个 65 536 × 4 ROM 扩展为一个 65 536 × 8 ROM,用来解释字长的扩展

例 10.3　使用例 10.2 中的存储器形成 64k × 16 ROM。

解:在这种情况下,需要一个存储 65 536个 16 位字的存储器。需要四个 64k × 4 ROM 完成这个任务,如图 10.41所示。

相关问题:需要多少个 64k × 1 ROM 才能实现如图 10.41 所示的存储器。

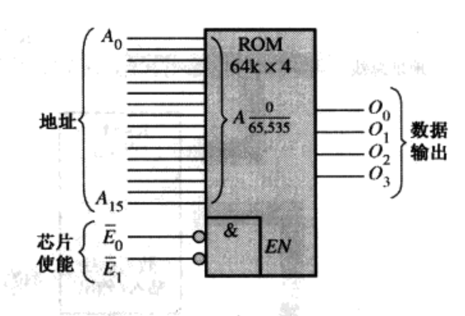


图 10.39　一个 64k × 4 ROM

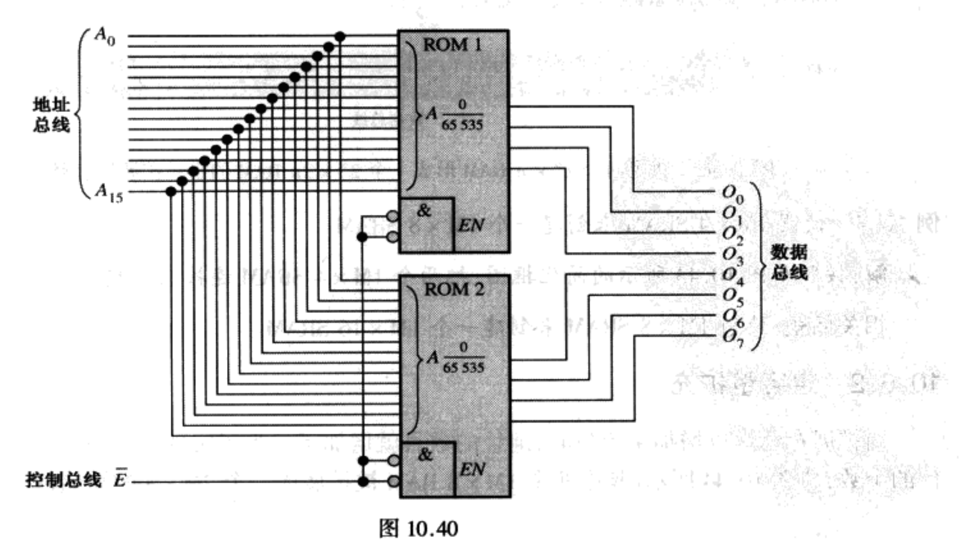


图 10.40

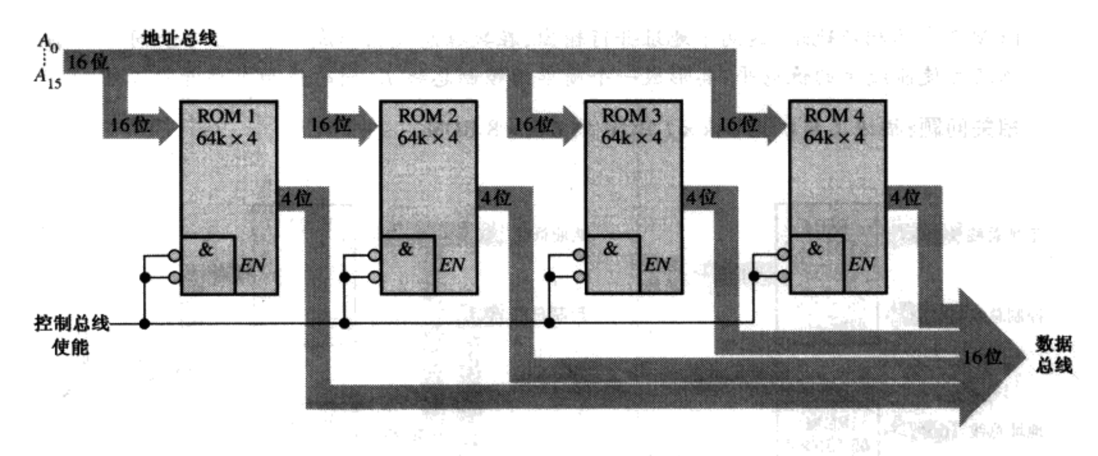


图 10.41

ROM 只有数据输出,但是 RAM 同时具有数据输入和数据输出。对于 RAM(SRAM 或者 DRAM)的字长扩充,数据输入和数据输出构成数据总线。因为数据输入和数据输出都使用相同的线。所以需要三态缓冲器。大多数 RAM 提供了内部三态电路。图 10.42 解释了 RAM 的扩展,以增加数据字长。

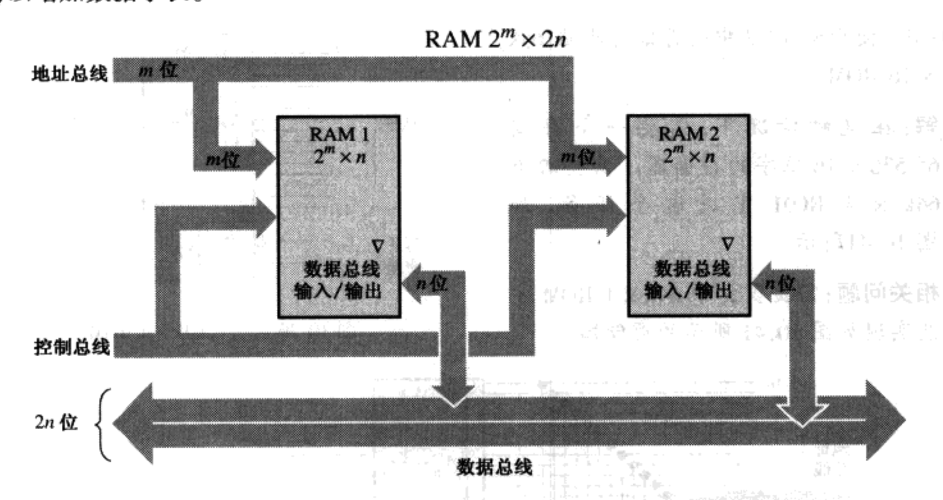


图 10.42 使用两个 $2^m \times n$ RAM 形成一个 $2^m \times 2n$ RAM 的字长扩展的解释

例 10.4 使用 1M×4 SRAM 来创建一个 1M×8 SRAM。

解:按照如图 10.43 所示的简化框图,把两个 1M×4 SRAM 连接在一起。

相关问题:使用 1M×8 SRAM 来创建一个 1M×16 SRAM。

10.6.2 字容量扩充

当扩展存储器以增加字容量时,地址的数目就增加了。为了实现这个增加,必须增加地址位的个数,如图 10.44 所示(其中两个 1M×8 RAM 被扩展成一个 2M×8 的存储器)。

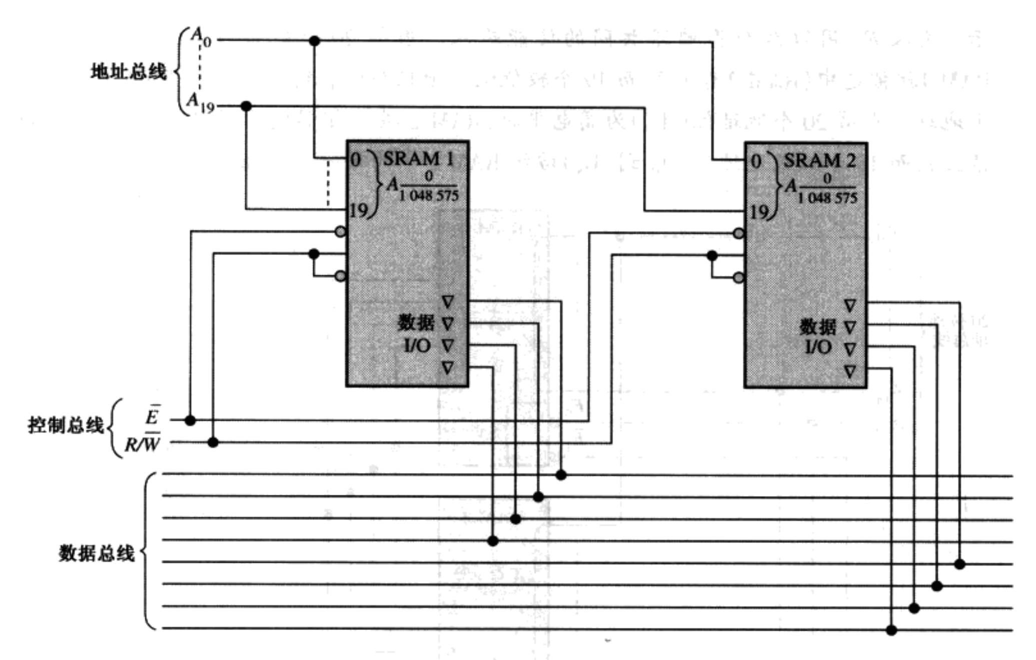
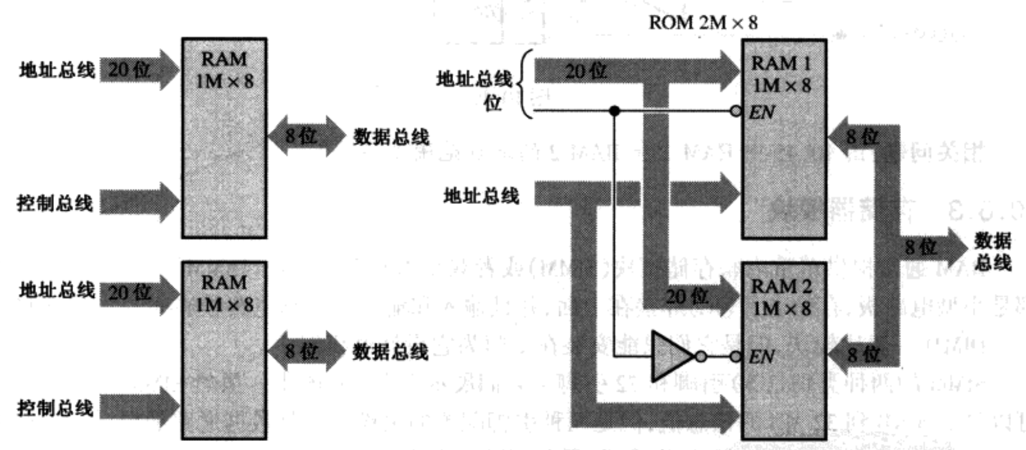


图 10.43



(a) 单个内存，每个存储 1 048 576 个 8 位字　　(b) 存储器被扩展形成需要 21 位地址总线的 2 M×8 RAM

图 10.44　字容量扩展的解释

每个独立的存储器都具有 20 个地址位以选择它的 1 048 576 个地址，如图 10.44(a)所示。扩展了的存储器具有 2 097 152 个地址，所以需要 21 个地址位，如图 10.44(b)所示。第 21 个地址位用来使能合适的存储芯片。扩展存储器的数据总线保持 8 位的宽度。扩充的细节如例 10.5所示。

例 10.5　使用 512k×4 RAM 来实现一个 1M×4 存储器。

解：扩展寻址通过连接芯片使能($\bar{E}_0$)输入到第 20 个地址位(A_{19})来实现，如图 10.45 所

示。输入 $\bar{E}_1$ 用做两个存储器共同的使能输入。当第 20 个地址位(A_{19})为低电平时，RAM 1就被选中(RAM 2 禁止)，而 19 个较低级的地址位(A_0 到 A_{18})访问 RAM 1 中的每一个地址。当第 20 个地址位(A_{19})为高电平时，RAM 2 被反相器输出的低电平使能(RAM 1 禁止)，而 19 个低级地址位(A_0 到 A_{18})访问 RAM 2 中的每一个地址。

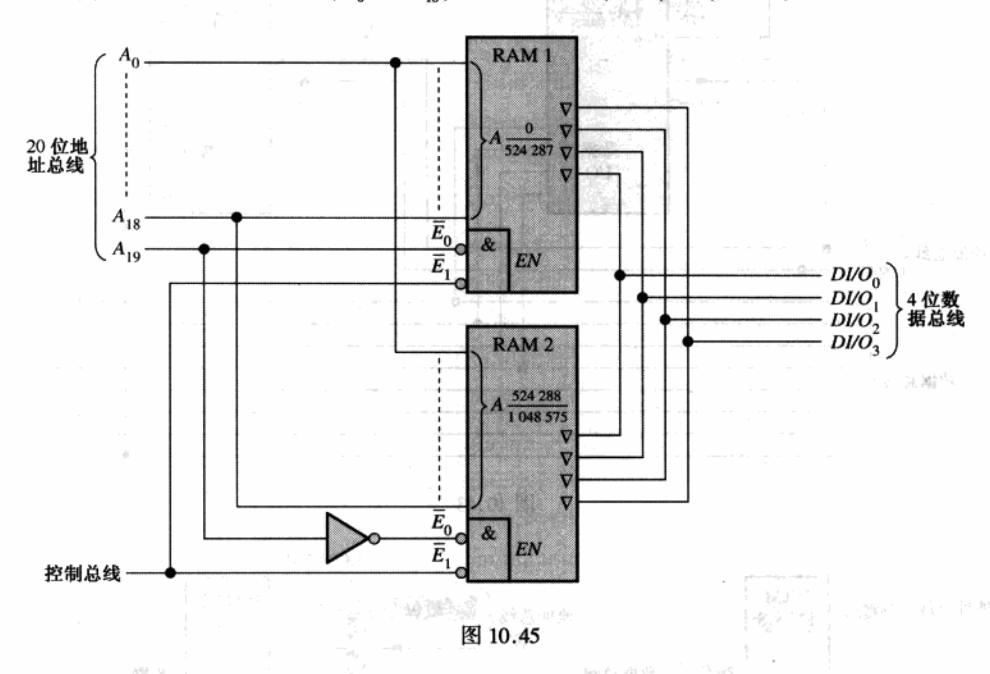


图 10.45

相关问题：图 10.45 中 RAM 1 和 RAM 2 的地址范围是多少？

10.6.3 存储器模块

RAM 通常提供单重内联存储模块(SIMM)或者双重内联存储模块(DIMM)。SIMM 和 DIMM 都是小型电路板，存储芯片(IC)焊接在上面，并且输入和输出连接到电路板底面边沿的连接口上。DIMM 一般比较快，但是它们只能安装在专门为它设计的机器中。

SIMM 的两种类别是 30 引脚和 72 引脚。它们展示在图 10.46 中。虽然 SIMM 的存储容量可以是 256 KB 到 32 MB 的任意值，但是两种引脚配置的关键区别是数据通路的尺寸。一般来说，30 引脚 SIMM 用于 8 位数据总线，如果处理更多的数据位，就需要更多的 SIMM。72 引脚 SIMM 用于 32 位数据总线，所以对于 64 位的数据总线来说，就需要一对 SIMM。

DIMM 看起来和 SIMM 很相似，但是在物理尺寸上仅仅是相当微小的增加，就会增加存储器的密度。关键区别是 DIMM 把输入和输出分布在 PC 主板的两侧，而 SIMM 只使用一侧。常见的 DIMM 配置为 72 引脚、100 引脚、144 引脚和 168 引脚，适用于 32 位和 64 位的数据通路。一般来说，DIMM 的容量范围从 4 MB 到 512 MB。

SIMM 和 DIMM 插入系统主板上的插座板上，如图 10.47 所示，一般会有几个插座用于存储器的扩展。当然，SIMM 和 DIMM 的插座是不同的，不可互相交换。

另一种标准的存储模块与 DIMM 相似，但是具有更高的总线速度，就是 RIMM(内存总线内

联存储模块)。同样,许多笔记本电脑使用一种 DIMM 的变异,称为 SODIMM,它的尺寸更小,有 144 个引脚和最多 256 MB 的容量。

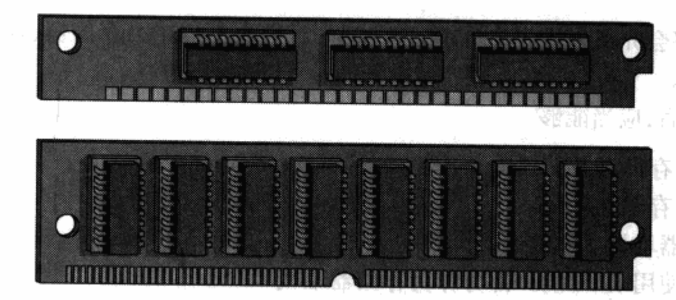


图 10.46 30 引脚和 72 引脚引的 SIMM

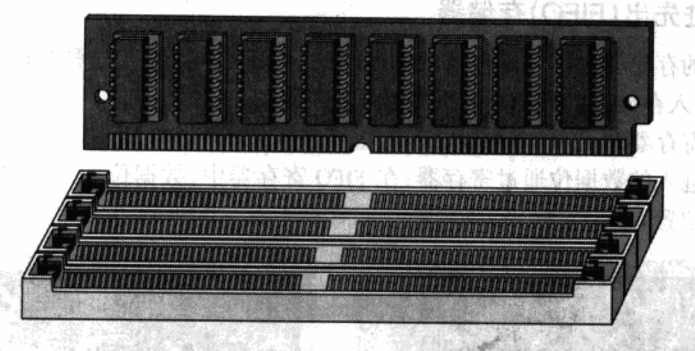


图 10.47 插入系统主板插座的 SIMM/DIMM

相应提示 存储器组件对静电极度敏感。当处理诸如 SIMM 和 DIMM 这样的存储芯片时,使用下面的预防措施:

- 在处理之前,通过触摸接地的表面或者带上含有高电阻的接地手腕皮带,以便放走身体上的静电。一个方便且可以可靠的接地方法是交流输出端接地;
- 在准备安装之前,不要把组件从它们的防静电袋子中取出;
- 不要把组件部分放在防静电袋子上,因为只有袋子内部是防静电的;
- 当处理 SIMM 或者 DIMM 时,抓住边沿或者金属安装托架。不要触摸主板上的元件或者边沿连接口的引脚;
- 绝不要在任何类型的表面上滑动元件的任何部分;
- 在工作区域内避免塑料、乙烯基、泡沫聚苯乙烯和尼龙。

安装 SIMM 和 DIMM 时,遵循以下的步骤:

1. SIMM 或 DIMM 板上的槽口和存储器插座上的槽口对齐;
2. 用力推压模块直到可靠地插在插座上面;
3. 通常,当模块完全插入时,插座两边的锁口会突然锁定位置。这些锁口也可以释放模块,这样模块就可以从插座上取出。

10.7 特殊类型的存储器

在本节中,将会介绍先进先出(FIFO)存储器、后进先出(LIFO)存储器、存储器堆栈和电荷耦合器件存储器。

学完本节以后,应当能够

- 描述 FIFO 存储器
- 描述 LIFO 存储器
- 讨论存储器堆栈
- 解释怎样使用 RAM 的一部分作为存储器堆栈
- 描述基本的 CCD 存储器

10.7.1 先进先出(FIFO)存储器

这种类型的存储器由移位寄存器的排列组成。术语 FIFO 是指这种存储器的基本操作,在操作中,首先写入存储器的数据位首先被读出。

常规移位寄存器和 FIFO 寄存器之间的重要区别如图 10.48 所示。在常规寄存器中,数据位仅仅作为新进入的数据位通过寄存器;在 FIFO 寄存器中,数据位即刻穿过寄存器而到达最右边空闲位的位置。

常规移位寄存器

输入	X	X	X	X	输出
0	0	*X*	*X*	*X*	⟶
1	1	0	*X*	*X*	⟶
1	1	1	0	*X*	⟶
0	0	1	1	1	⟶

X=未知数据位

在常规移位寄存器中,数据停留在左边,直到有添加的数据"强迫"它通过寄存器

FIFO 移位寄存器

输入	—	—	—	—	输出
0	—	—	—	0	⟶
1	—	—	1	0	⟶
1	—	1	1	0	⟶
0	0	1	1	0	⟶

— = 空闲位置

在 FIFO 移位寄存器中,数据"陷"入(靠右进入)寄存器

图 10.48 常规移位寄存器 FIFO 移位寄存器的比较

图 10.49 是 FIFO 串行存储器的框图。这种特殊的存储器有四个串行 64 位数据寄存器和一个 64 位控制寄存器(标记寄存器)。当数据在移入脉冲的作用下进入寄存器时,它们在标记寄存器的控制下自动移向最靠近输出的空位置。数据不能进入被占用的位置,但是,当数据位在移出脉冲的作用下移出时,寄存器中剩余的数据位自动向输出方向移到下一个位置。在异步 FIFO 中,由于使用两个独立的时钟,数据的移出独立于数据进入。

10.7.2 FIFO 应用

FIFO 寄存器的一个重要应用领域就是这样的情况,两个不同数据速率的系统必须进行通信。数据可以在一种速率下进入 FIFO 寄存器,而在另一种速率下从寄存器中取出。图 10.50 给出了在这些情况下 FIFO 寄存器是如何使用的。

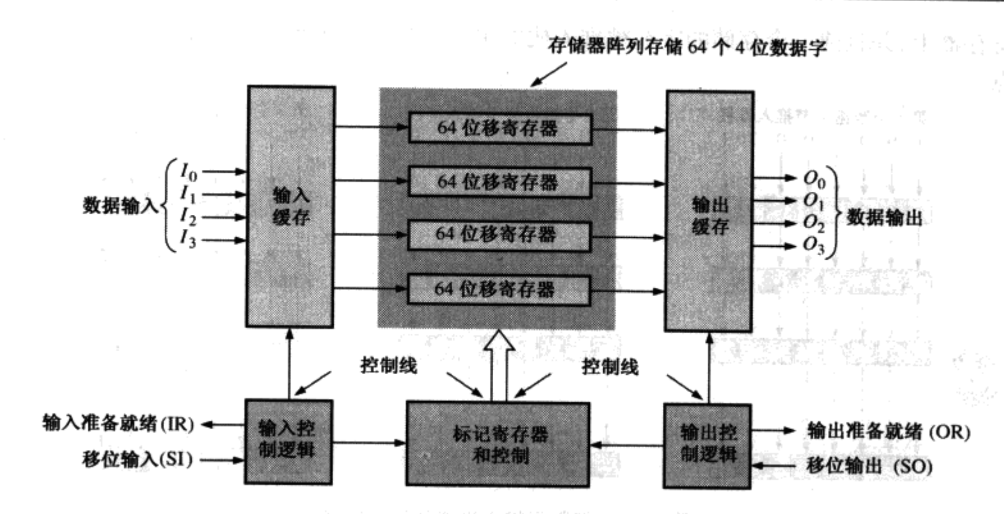


图 10.49 典型的 FIFO 串行存储器的框图



(a) 不规则遥测数据可以被存储下来，并且以一个恒定的速率再次传送

低速率数据 → FIFO寄存器 → 高速率数据

(b) 数据输入可以在一个很低的键盘速率下存储下来，然后以一个用于处理的高速率传送

恒速率数据 → FIFO寄存器 → 脉冲串数据

(c) 数据输入可以在恒定速率下存储起来，然后以脉冲串的形式输出

脉冲串数据 → FIFO寄存器 → 恒速率数据

(d) 脉冲串数据可以存储下来，并且重新格式化成恒定速率的数据

图 10.50 在数据速率缓存应用中的 FIFO 寄存器

10.7.3 后进先出(LIFO)存储器

LIFO(后进先出)存储器经常出现在涉及微处理机和其他计算系统的应用中。它允许数据存储下来，然后以相反的顺序调用；也就是说，最后存储的数据字节是最先取出的数据字节。

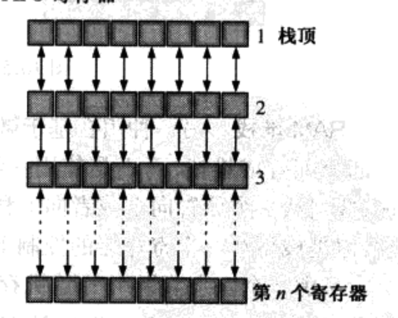


图 10.51 寄存器堆栈

寄存器堆栈 LIFO 存储器一般是指下推堆栈。在一些系统中，它一般由如图 10.51 所示的寄存器组来实现。一个堆栈可以由任意数目的寄存器组成，但是顶部的寄存器一般称为栈顶。

为了解释这个原理，一个字节的数据被并行推入到栈顶上。每一个相继的字节都把前一个字节推入到下一个寄存器中。这个过程如图 10.52 所示。注意新的字节常常总是推入到顶

部寄存器上,并且前一个存储的字节被推入栈的更深一层。下推堆栈这个名称就来自于这个特点。

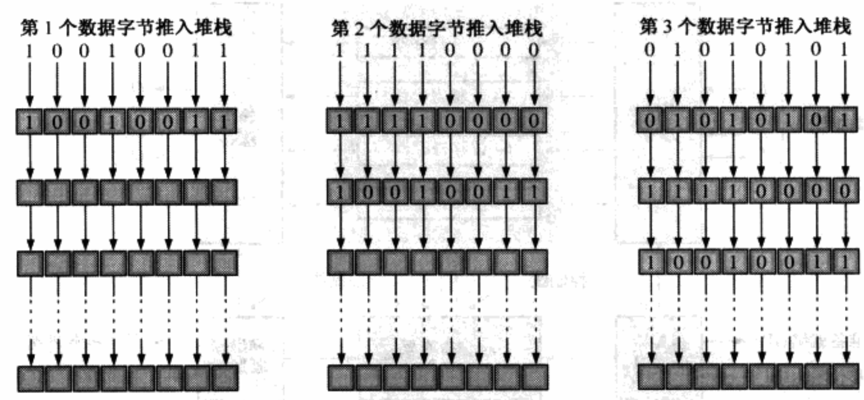


图 10.52 把数据推入堆栈的简单解释

数据字节以相反的顺序被取出。最后进入的数据字节总是位于栈顶,所以当这个数据从堆栈中取出时,其他的字节就会跳到高一级的位置。这个过程如图 10.53 所示。

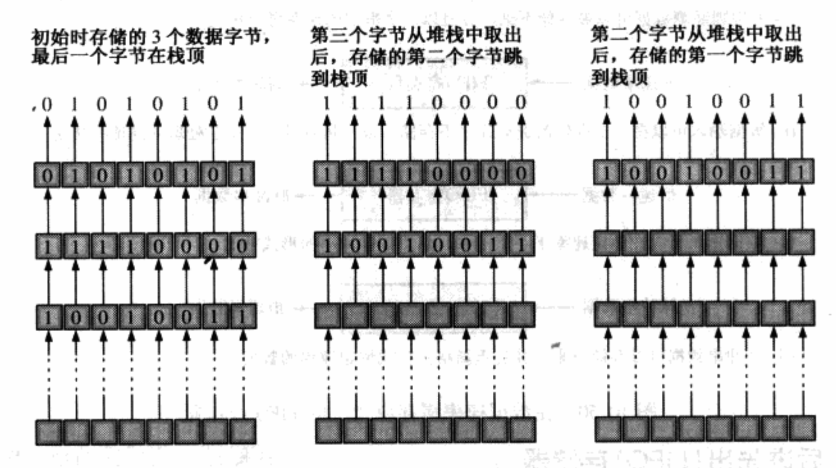


图 10.53 把数据从堆栈取出的简单解释

RAM 堆栈 另一种用在基于微处理器的系统中实现 LIFO 存储器的方法就是把 RAM 的一部分配置成堆栈,而不是使用一组专门的寄存器。正如所看到的那样,对于一个寄存器堆栈,数据从一个位置向上或者向下移动到下一个相邻位置。在 RAM 堆栈中,数据本身并不移动,但是栈顶在一个寄存器的控制下移动,这个寄存器称为堆栈指针。

考虑一个以字节组织的随机存储器——也就是每个地址包含 8 个位,如图 10.54 所示。例如,二进制地址 0000 0000 0000 1111 在十六进制中可以写成 000F。一个 16 位地址的最小十六进制数值为 0000_{16},而最大数值为 $FFFF_{16}$。使用这种表示方法,一个 64 kB 字节的存储阵列可以表示为如图 10.54 所示的阵列。最低存储地址是 0000_{16},而最高内存地址为 $FFFF_{16}$。

现在,考虑留出 RAM 的一部分用做堆栈。堆栈指针这个特殊的独立寄存器,包含栈顶的

地址,如图10.55所示。一个4位数十六进制表达用来表示二进制地址。在图中,为了解释的目的,我们选择了一个地址。

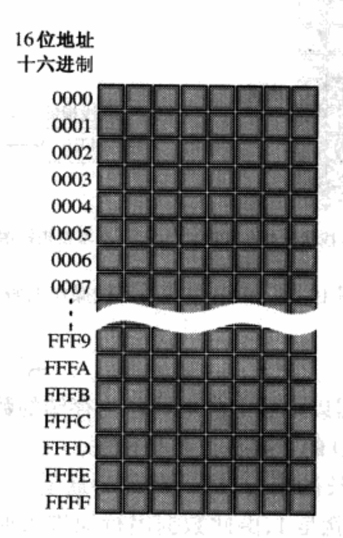
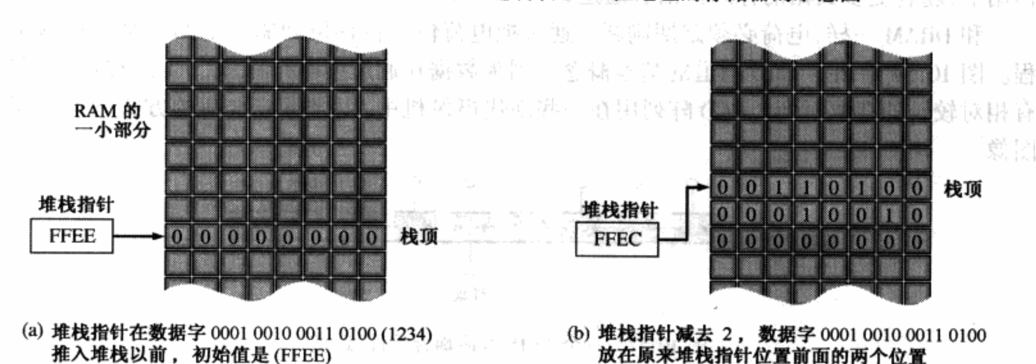


图10.54 一个64 KB字节、用十六进制表示16位地址的存储器的示意图



(a) 堆栈指针在数据字0001 0010 0011 0100 (1234)推入堆栈以前,初始值是(FFEE)

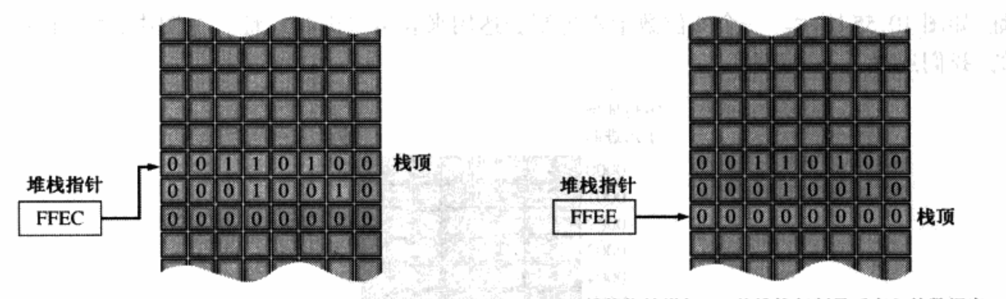
(b) 堆栈指针减去2,数据字0001 0010 0011 0100放在原来堆栈指针位置前面的两个位置

图10.55 RAM堆栈推入操作的解释

现在看看数据是怎样被推入堆栈中的。堆栈指针初始时位于地址$(\text{FFEE})_{16}$处,就是如图10.55(a)所示的栈顶。堆栈指针随后被减去(减少)2,到达地址$(\text{FFEC})_{16}$。这就从栈顶移到了较低的存储地址,如图10.55(b)所示。注意栈顶和固定寄存器堆栈的栈顶不一样,它不是固定不变的,而是随着数据的存储在RAM中向下移动(移向较低的地址)。图10.55(b)解释了两个字节(一个数据字)随后被推入了堆栈中。在数据字被存储之后,栈顶位于$(\text{FFEC})_{16}$。

图10.56给出了RAM堆栈的出栈(POP)操作。堆栈中最后存入的一个数据首先被读出。位于地址处的堆栈指针FFEC增加2,到达地址$(\text{FFEE})_{16}$,出栈(POP)操作的执行如图10.56(b)所示。记住在读操作时,RAM是非破坏性的,所以数据字在出栈操作之后仍然保持在存储器中。只有当写入新的字覆盖数据字时,这个数据字才会被破坏。

RAM堆栈可以有任意的深度,这取决于为实现这个目的所分配的连续的存储地址数。



(a) 从堆栈复制（出栈）数据字前，堆栈指针处在(FFEC)　(b) 堆栈指针增加 2，从堆栈复制最后存入的数据字

图 10.56 RAM 堆栈的出栈操作的解释

10.7.4 CCD 存储器

CCD(电荷耦合器件)存储器以电容上的电荷作为存储的数据。然而,和动态 DRAM 不同,其存储单元不包括晶体管。CCD 的主要优点是高密度。

CCD 存储器由一行一行较长的半导体电容(称为通道)组成。存储在电容上的电荷较少时,就是 0,存储的电荷较多时,就是 1,由此数据串行地进入通道。这些电荷包在时钟信号的作用下,随着更多数据的存入而沿着通道移位。

和 DRAM 一样,电荷必须定期刷新。通过把电荷包串行移位到刷新电路中来完成这个过程。图 10.57 给出了 CCD 通道的基本概念。因为数据在通道中串行移位,所以 CCD 存储器具有相对较长的存取时间。CCD 阵列用在一些现代摄像机中,以光感应电荷的方式来捕捉视频图像。

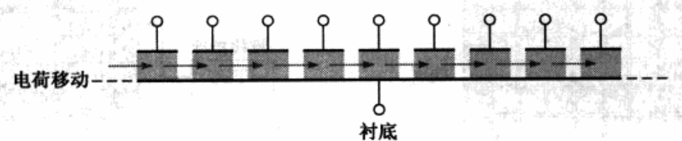


图 10.57 一个 CCD(电荷藕合器件)通道

自测题 (答案在本章的结尾)

1. 一个存储器有 1024 个地址,每个地址可以存储 8 个位,那么它的位容量为
   (a) 1024　(b) 8192　(c) 8　(d) 4096
2. 一个 32 位数据字由下面哪一项组成?
   (a)2 字节　(b)4 个半字节　(c)4 个字节　(d)3 个字节和 1 个半字节
3. 数据在什么时候存储在随机存储器(RAM)中?
   (a)读操作　(b)使能操作　(c)写操作　(d)寻址操作
4. 随机存储器(RAM)中给定地址中存储的数据在什么情况下会丢失?
   (a)电源关闭　(b)数据从地址中读出
   (c)在地址中写入新数据　(d)答案(a)和(c)
5. 一个 ROM 是
   (a)非易失性存储器　(b)易失性存储器
   (c)读/写存储器　(d)以字节组织的内存

6. 具有 256 个地址的存储器有
   (a)256 条地址线 (b)6 条地址线
   (c)1 条地址线 (d)8 条地址线
7. 以字节组织的存储器有
   (a)1 条数据输出线 (b)4 条数据输出线
   (c)8 条数据输出线 (d)16 条数据输出线
8. SRAM 中的存储单元是
   (a)触发器 (b)电容 (c)熔丝 (d)磁性区域
9. 一个 DRAM 必须
   (a)定期置换 (b)定期刷新 (c)一直使能 (d)在每次使用之前编程
10. 闪存是
   (a)易失的 (b)只读存储器 (c)读/写存储器
   (d)非易失的 (e)答案(a)和(c) (f)答案(c)和(d)

习题

10.1 节 半导体存储器基础

1. 识别图 10.58 中的 ROM 和 RAM。

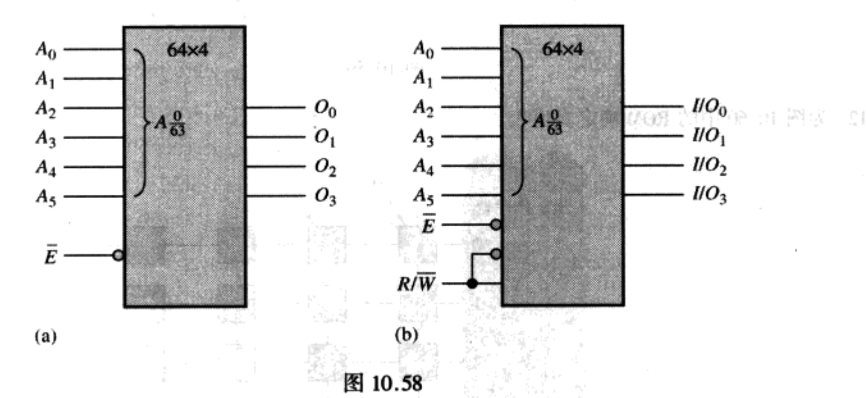


图 10.58

2. 解释 RAM 和 ROM 为什么都是随机存储器。
3. 解释地址总线和数据总线的用途。
4. 下面每个十六进制数分别表示哪个存储地址(0 到 256):
   (a) $0A_{16}$ (b) $3F_{16}$ (c) CD_{16}

10.2 节 随机存储器(RAM)

5. 一个静态存储器阵列具有 4 行,并且相似于图 10.9 中的存储器,其初始时存储的都是 0。在下面的条件之后,它的内容是什么?假设一个 1 选择一行。
   行 0 = 1,数据进入(位 0) = 1
   行 1 = 0,数据进入(位 1) = 1
   行 2 = 1,数据进入(位 2) = 1
   行 3 = 0,数据进入(位 3) = 0
6. 为 512 × 8 位的静态 RAM 绘制基本逻辑图,给出所有的输入和输出。

7. 假设某个 64k×8 SRAM 的结构相似于图 10.11 中的 SRAM,确定它的存储单元阵列中的行和 8 位列的数目。
8. 为一个 64k×8 存储器重新绘制图 10.11 中的模块图。
9. 解释 SRAM 和 DRAM 之间的区别。
10. 具有 12 条地址线的 DRAM,其容量是多少?

10.3 节　只读存储器(ROM)

11. 对于图 10.59 中的 ROM 阵列,确定所有可能输入组合的输出,并以表格的形式把它们总结出来(深色单元为 1,浅色单元为 0)。

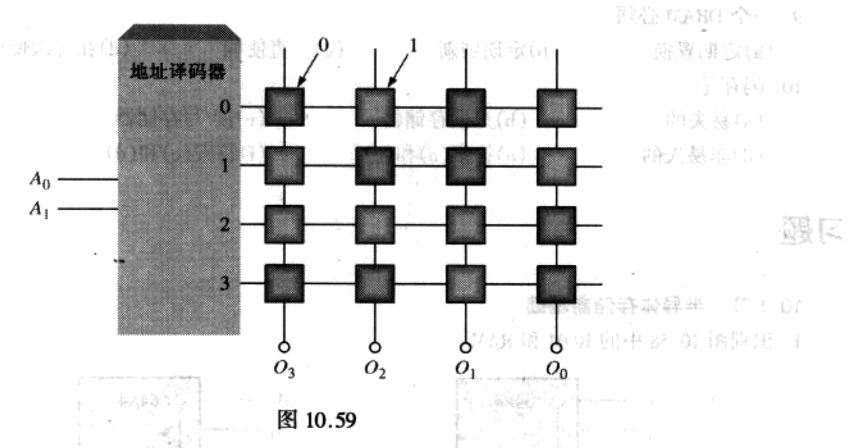


图 10.59

12. 为图 10.60 中的 ROM 确定真值表。

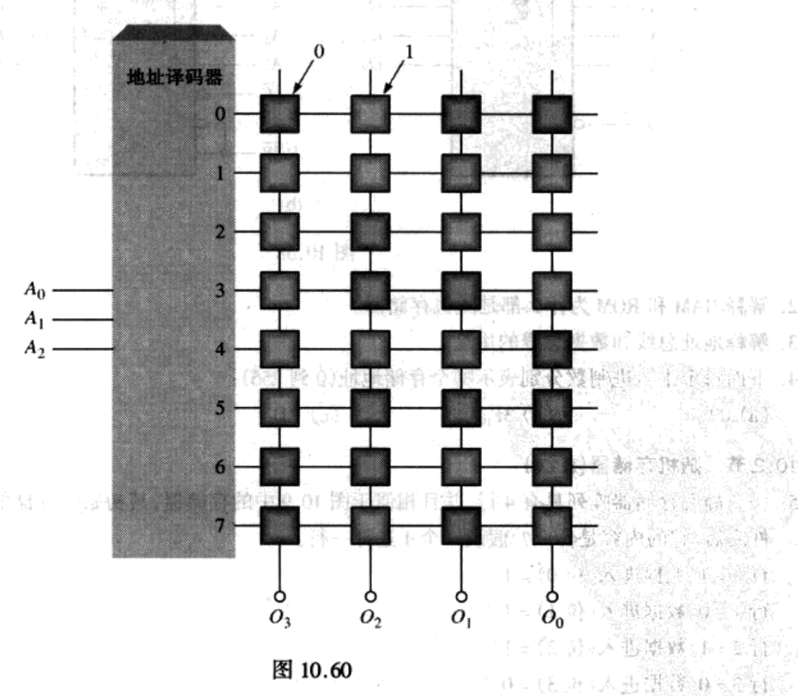


图 10.60

13. 使用类似于例 10.1 的过程，设计一个 ROM 用以将单个数字 BCD 变换为余 3 码。
14. 某个 ROM 具有 14 条地址线和 8 个数据输出，那么其总位容量是多少？

10.6 节 存储器扩展

15. 使用 16k×4 的 DRAM 来构建一个 64k×8 的 DRAM。给出逻辑图。
16. 使用框图，给出怎样扩展 64k×1 的动态 RAM，以构建一个 256k×4 RAM。
17. 习题 15 和习题 16 中存储器的字长和字容量是多少？

10.7 节 特殊类型的存储器

18. 完成图 10.61 中的时序图，给出初始为 0 的 FIFO 串行存储器的输出波形和图 10.49 中所给出的存储器的输出波形相似。

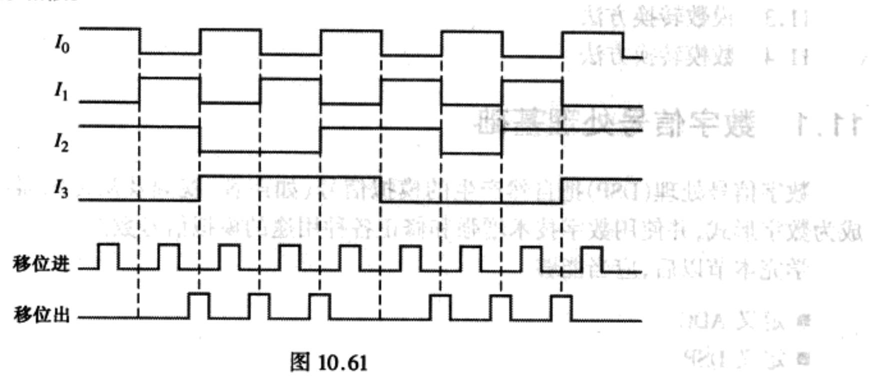


图 10.61

19. 考虑一个 4096×8 RAM，其中最后 64 个地址用做 LIFO 堆栈。如果 RAM 中的第一个地址是 000_{16}，指出用做堆栈的这 64 个地址。
20. 在习题 19 的存储器中，16 个字节被推入堆栈中。第一个字节所处的地址是什么？最后一个字节所处的地址是什么？

自测题答案

1. (b) 2. (c) 3. (c) 4. (d) 5. (a) 6. (d) 7. (c) 8. (a) 9. (b) 10. (f)

第 11 章　数字信号处理

章节提纲

11.1　数字信号处理基础

数字信号处理(DSP)把自然产生的模拟信号(如声音、视频及从传感器产生的信息)转换成为数字形式,并使用数字技术增强和修正各种用途的模拟信号数据。

学完本节以后,应当能够

- 定义 ADC
- 定义 DSP
- 定义 DAC
- 绘制出数字信号处理系统的基本框图

数字信号处理系统首先把连续变化的模拟信号转换成一系列离散的电平。这些电平随着模拟信号的变化,就像楼梯一样,如图 11.1 所示的正弦波的情况。把原始模拟信号变成近似"楼梯"的过程是由采样和保持电路完成的。

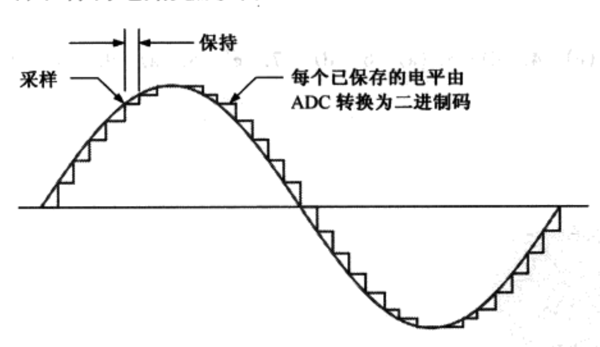


图 11.1　原始模拟信号(正弦波)及它的近似"楼梯"的形状

下一步,近似"楼梯"形状的波形被一个称为模数转换(A/D)的过程量化成为二进制码,这种二进制编码表示"楼梯"中的每一个离散的台阶。完成 A/D 转换的电路就是一个模数转换器(ADC)。

一旦模拟信号转换成为二进制码的形式,就会加到一个数字信号处理器上。DSP 可以对进入的数据完成各种操作,如除去不需要的干扰,增加某个信号频率的幅度,而对其他的信号

进行衰减;对数据编码以便可靠的传输;以及检测和纠正传输代码过程中的错误。DSP 还使其他许多工作成为可能,比如清除声音记录,消除来自通信线路的回波;增强来自 CT 扫描的图像,以得到更好的医学诊断;对便携式电话的通话进行编码以保护隐私。

经过 DSP 处理的信号还可以转换回模拟信号,而且比原来的模拟信号有了很大改善。这个过程是由数模转换器(DAC)完成。图 11.2 显示了一个典型的数字信号处理系统的基本框图。

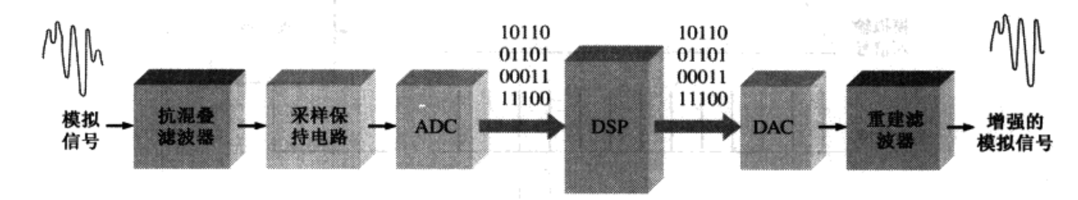


图 11.2 数字信号处理系统的基本框图

DSP 实际上是一种特殊类型的微处理器,但是它在许多重要的方面不同于一般用途的微处理器。一般情况下,微处理器用于一般用途的操作,并且由大型的软件包操作。DSP 也可用于一些特殊应用,通常它们是快速数字处理器,必须以实时方式处理信息,而且使用专门的算法(程序)。系统中的模数转换器(ADC)必须对进入的模拟数据进行采样,通常采样点要足够多,以便获取和信号幅度波动有关的全部信息。DSP 的计算速度必须和接收到的采样数据速率一样快,以便与 ADC 的采样速率保持步调一致。一旦数字数据经过 DSP 的处理,就会进入数模转换器(DAC),以转换回模拟形式。

11.2 模拟信号转换为数字信号

为了使用数字技术处理信号,输入的模拟信号必须转换成为数字形式。

学完本节以后,应当能够

- 解释把模拟信号转换为数字信号的基本过程
- 描述采样保持功能的用途
- 定义奈奎斯特(Nyquist)频率
- 解释产生混叠信号的原因,并讨论消除它的方法
- 描述 ADC 的用途

11.2.1 采样和滤波

图 11.2 的系统图中的前两个方框是抗混叠滤波器及采样保持电路。采样保持功能完成两个操作,第一是采样。采样就是在一个波形上获取足够数量的离散值,这些值将描述这个波形形状的特征。采样值越多,描述的波形就越准确。采样把模拟信号转换成为一系列脉冲,每个脉冲表示信号在某个给定时刻的振幅。采样过程如图 11.3 所示。

模拟信号经过采样后,必须满足一定的标准,以便准确地表达原始信号。所有的模拟信号(除了纯粹的正弦波)都包含一些频率组成的频谱,称为谐波。模拟信号的谐波是一些不同频率和振幅的正弦波。一个给定周期波的谐波加在一起,结果就得到原始信号。在对信号采样前,必须经过一个低通滤波器(抗混叠滤波器),以消除超过确定值谐波频率,这个确定值取决于奈奎斯特频率。

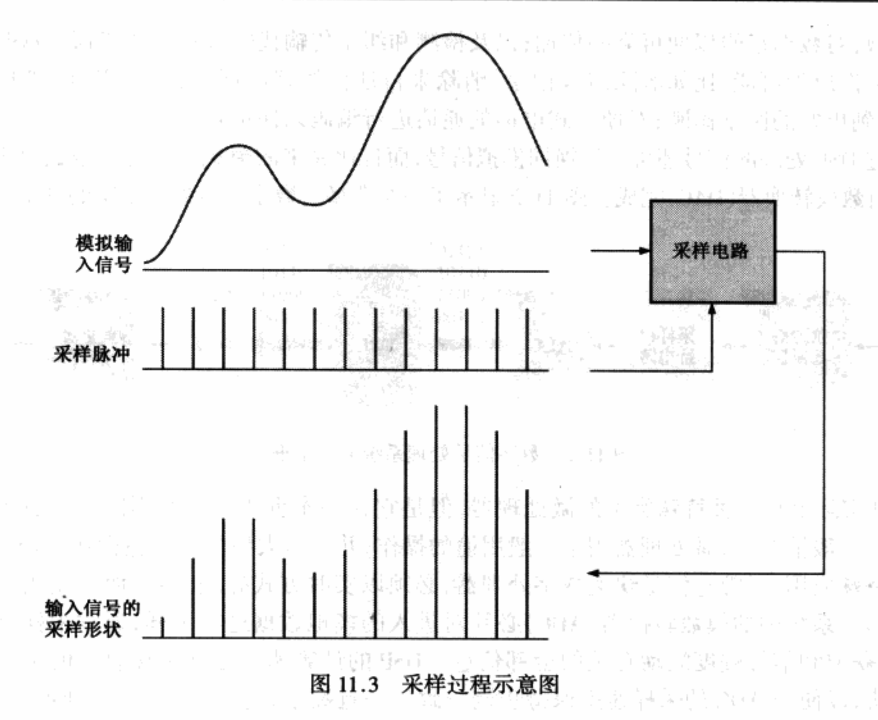


图 11.3　采样过程示意图

采样定理　注意图 11.3 中有两个输入波形。一个是模拟信号,另一个是采样脉冲波形。采样定理的陈述是,要表示一个模拟信号,采样频率 f_{sample} 必须至少是模拟信号最高的频率成分 $f_{a(max)}$ 的两倍。另一种表达方式是,最高模拟频率不能高于采样频率的一半。频率 $f_{a(max)}$ 称为"奈奎斯特频率",由式(11.1)表示。在实际情况下,采样频率应当高于最高模拟频率的两倍:

$$f_{sample} \geqslant 2f_{a(max)} \tag{11.1}$$

为了直观地理解采样定理,一个简单的"弹跳球"的比喻很有用。尽管它不能十分完美地表达电子信号的采样过程,但它确实能够说明其基本想法。如果一个球在弹跳期间的一个瞬间被照相(采样),如图 11.4(a)所示,这时不能说出任何有关这个球路径的情况,除了说出它离开地板以外,说不出这个球向上、向下或弹跳的高度。如果在一次弹跳期间在两个相等时间间隔的瞬间拍照,如图 11.4(b)所示,那么只能得到关于球运动情况的最少的信息量,而不知道球弹跳的距离。在这种特殊情况下,仅知道在两次拍照的时间里球是在空中的,球弹跳的最大高度比每张照片中的高度都高或至少相等。如果拍了四张照片,如图 11.4(c)所示,那么球在一次弹跳期间所经过的路径开始呈现出来。拍的照片(采样)越多,可以确定的球的弹跳路径就越精确。

滤波的需要　低通滤波对于消除模拟信号的谐波频率是十分必要的,谐波频率指所有超过奈奎斯特频率的那部分。如果模拟信号中的任何谐波频率超过了奈奎斯特频率,就会出现不想得到情况,称之为发生假信号混叠。假信号是在采样频率小于信号频率的两倍时产生的信号。假信号的频率小于被采样模拟信号的最高频率,因此在输入模拟信号的频谱或频带范围内造成失真。这样的信号实际上会"假装"成为模拟信号的一部分,但真实并非如此,因此称为"假信号"。

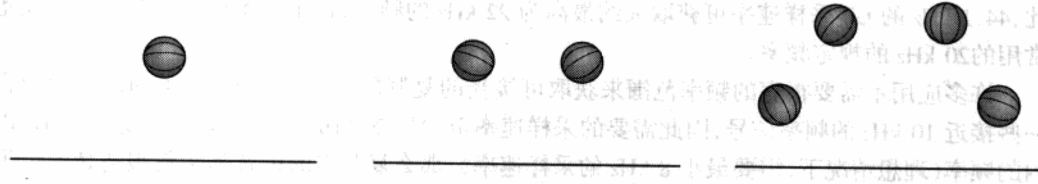


图 11.4　采样定理的弹跳球模拟

另一种观察混叠假信号的方法是把假信号看做采样脉冲产生一个高于和低于采样频率的谐波频谱，如图 11.5 所示。如果模拟信号包含高于奈奎斯特频率的频率，那么这些频率将与采样波形的频谱重叠，如图所示，从而产生干扰。采样波形中较低成分的频率和模拟波形的频谱混合，结果会产生一个信号混叠的错误。

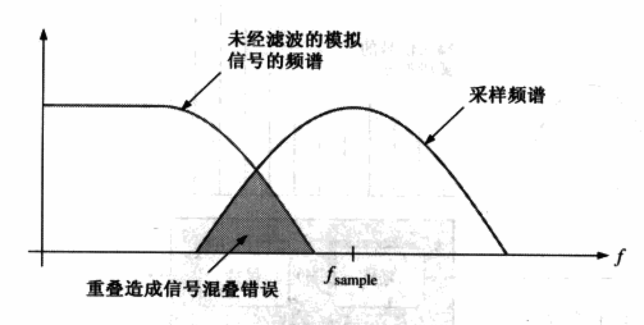


图 11.5　$f_{sample} < 2f_{a(max)}$ 情况的基本解释

必须使用低通抗混叠滤波器，以便对一个给定的采样频率限制模拟信号的频谱。为了避免假信号混叠错误，滤波器至少必须把所有超最小频率的模拟信号频率除掉，这个最小频率就是采样频谱中的最小频率，如图 11.6 所示。还可通过把采样频率提高到足够大的方法来避免假信号混叠错误。但是，最高采样频率通常会受到其后的模数转换器(ADC)性能的限制。

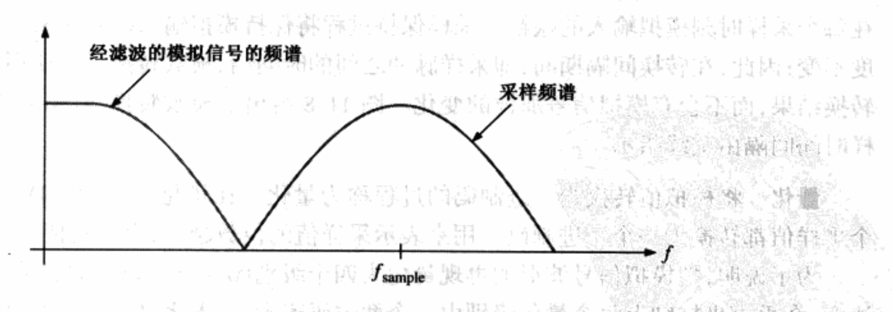


图 11.6　经过低通滤波，模拟信号和采样信号的频谱不会重叠，从而消除了假信号混叠错误

一个应用　采样的一个应用例子是在数字音频设备中。使用的采样速率为 32 kHz、44.1 kHz 或 48 kHz(每秒钟的采样次数)。48 kHz 的速率最常用，但 44.1 kHz 的速率常用于音频 CD 和预先录制好的磁带。根据奈奎斯特速率，采样频率至少必须是音频信号的两倍。因

此,44.1 kHz 的 CD 采样速率可获取大约最高为 22 kHz 的频率,这个频率超过了多数音频设备常用的20 kHz 的规定频率。

许多应用不需要很宽的频率范围来获取可接受的复制的声音。例如,人类的语音中包含一些接近 10 kHz 的频率信号,因此需要的采样速率至少是 20 kHz。但是,如果仅复制 4 kHz 以内的频率(理想情况下,需要最小 8 kHz 的采样速率),那么复制出的语音是听得很清楚的。另一方面,如果没有以一个足够高的速率对声音信号采样,那么假信号混叠的效果就会以背景噪声和失真的形式变得很明显。

11.2.2 保持采样值

如图 11.2 所示,保持操作是采样保持框图的一部分。经过滤波和采样,取得的值必须保持不变,直到下一次采样的发生。这是必须的,以便让 ADC 有时间去处理采样值。采样和保持操作的结果就是近似模拟输入信号波形的"楼梯"形状的波形,如图 11.7 所示。

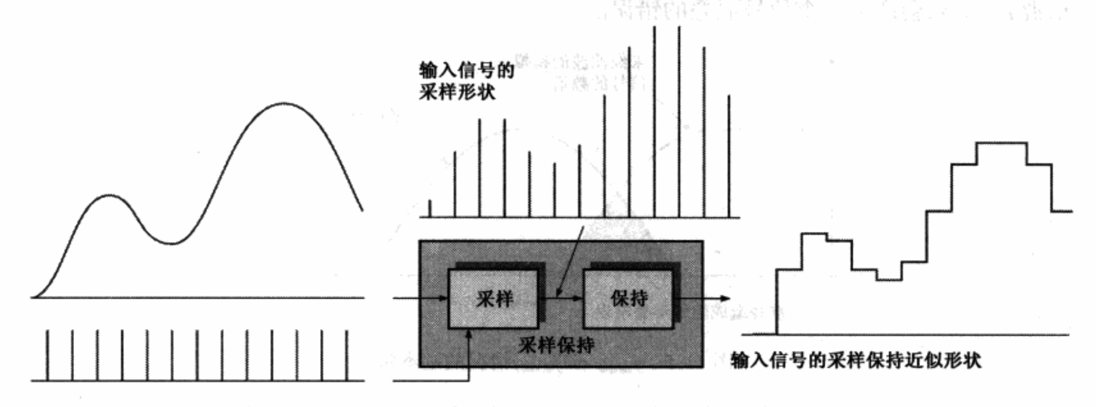


图 11.7 采样保持操作示意图

11.2.3 模数转换器

模数转换是把采样保持电路的输出转换成为一系列二进制码的过程,这些二进制码表示在每个采样时刻模拟输入的振幅。采样保持过程将保持模拟输入信号在两次采样脉冲间的幅度不变;因此,在转换间隔期间,即采样脉冲之间的时间内,模数转换可以使用一个常数值作为转换结果,而不会有模拟信号那样的变化。图 11.8 给出了模数转换器(ADC)的基本功能。采样时间间隔由虚线指示。

量化 将模拟值转换为二进制码的过程称为量化。在量化过程中,ADC 把模拟信号的每个采样值都转换为一个二进制码。用来表示采样值的位数越多,表示的精度就越高。

为了说明,把模拟信号波形的再现量化为四个级别(0~3)。如图 11.9 所示,需要两位码。注意,在垂直坐标轴上每个量化级别由一个两位码表示,沿着水平坐标轴的每个采样间隔都有编号。量化过程的总结如表 11.1 所示。

如果用得到的 2 位数字码重构原始波形,那么由数模转换器(DAC)来做,可以得到如图 11.10 所示的波形。如图所示,仅用两位码来表示采样值会使精度降低很多。

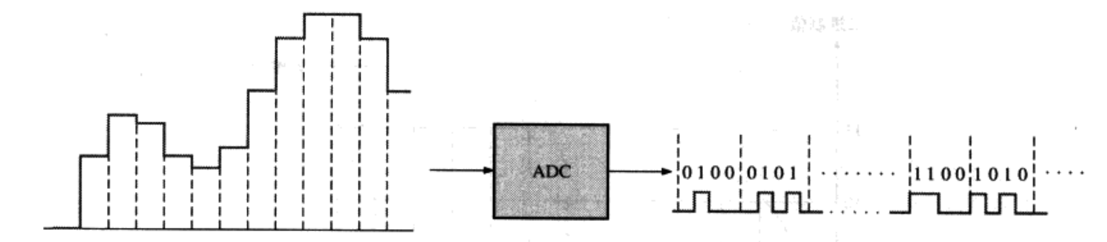


图 11.8 模数转换器(ADC)的基本功能(二进制码和位数仅仅是为了例证而任意选择的),表示ADC输出波形的二进制码也显示在图中

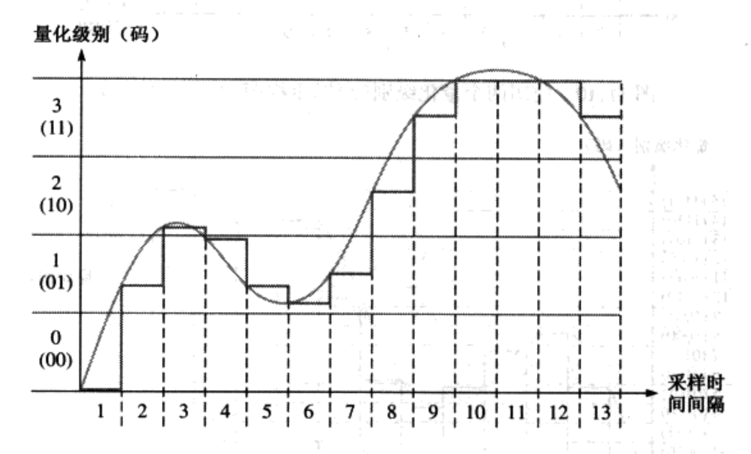


图 11.9 具有四个量化级别的采样保持输出波形,原始模拟波形以浅色表示作为参考

表 11.1 图 11.9 所示波形的两位量化码

采样时间间隔	量化级别	码
1	0	00
2	1	01
3	2	10
4	1	01
5	1	01
6	1	01
7	1	01
8	2	10
9	3	11
10	3	11
11	3	11
12	3	11
13	3	11

现在我们来看更多位的码是如何提高精度的。图 11.11 显示了用 16 个量化级别(4 位)来表示的同一波形。4 位量化过程总结如表 11.2 所示。

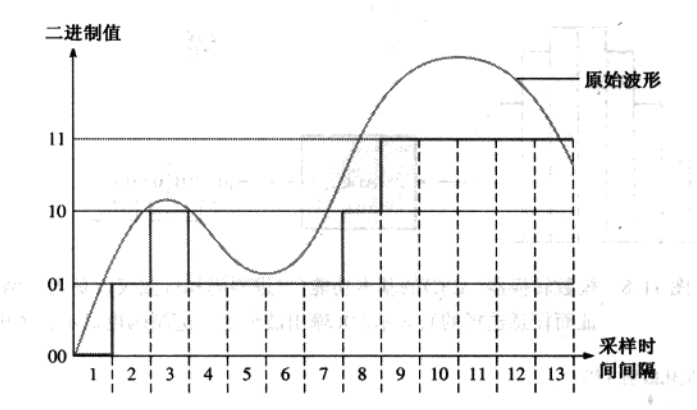


图 11.10 使用四个量化级别(2 位)重构图 11.9 中的波形

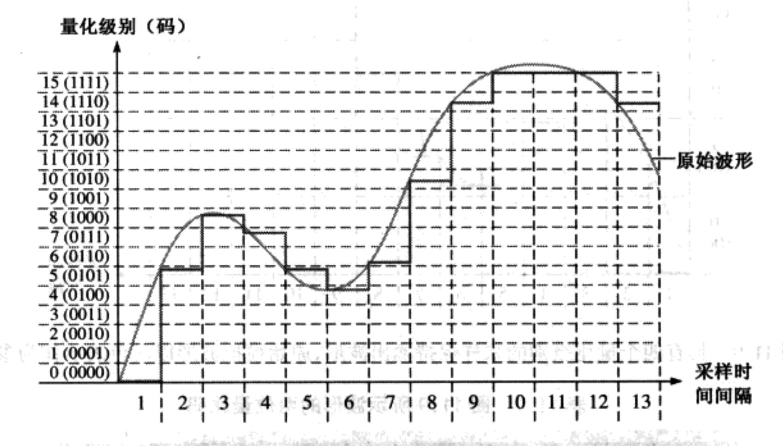


图 11.11 具有 16 个量化级别的采样保持输出波形

表 11.2 图 11.11 中波形的四位量化

采样时间间隔	量化级别	码
1	0	0000
2	5	0101
3	8	1000
4	7	0111
5	5	0101
6	4	0100
7	6	0110
8	10	1010
9	14	1110
10	15	1111
11	15	1111
12	15	1111
13	14	1110

如果用得到的4位数字码重构原始波形，就可以得到如图11.12所示的波形。如图所示，比起在图11.10中使用四个量化级别的情况，这次得到的结果更接近于原始波形。这表明用更多的量化位可得到更高的精度。多数集成电路ADC使用8位到24位，采样保持功能有时就包含在ADC芯片中。在接下来的小节中将介绍几种类型的ADC。

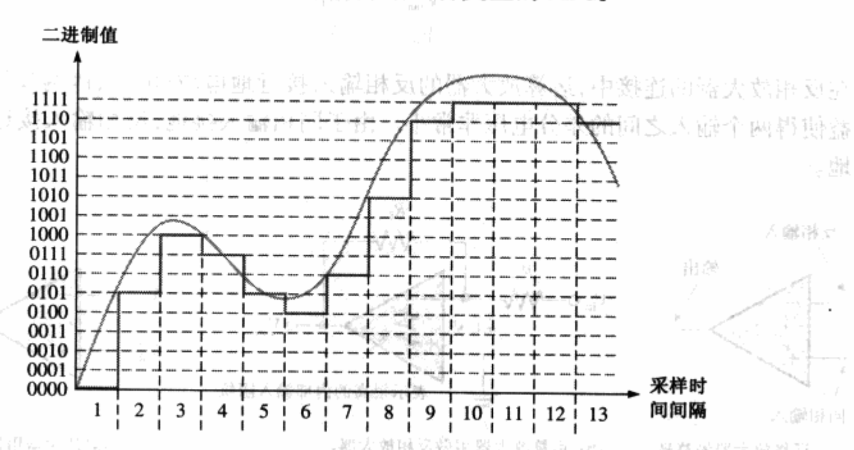


图11.12 使用16个量化级别(4位)重构图11.11中的波形，原始模拟波形以浅色表示作为参考

11.3 模数转换方法

如前所述，模数转换是把模拟量转换成数字形式的过程。测量得到的量必须以数字的形式处理、显示或存储，这一点很有必要。现在，我们来分析几种普通类型的模数转换器(ADC)。ADC的两个重要参数是分辨率(即量化的位数)和"通过量"[即采样速率，一个ADC能够处理的每秒采样(sps)的个数]。

学完本节以后，应当能够

- 从根本上解释什么是运算放大器
- 说明运算放大器如何用做反相放大器或比较器
- 解释快速ADC的工作
- 讨论双积分型ADC
- 描述逐次渐近型ADC的操作
- 描述delta-sigma ADC
- 讨论如何测试ADC，以寻找丢失的码、错误的码和偏差

11.3.1 快速浏览运算放大器

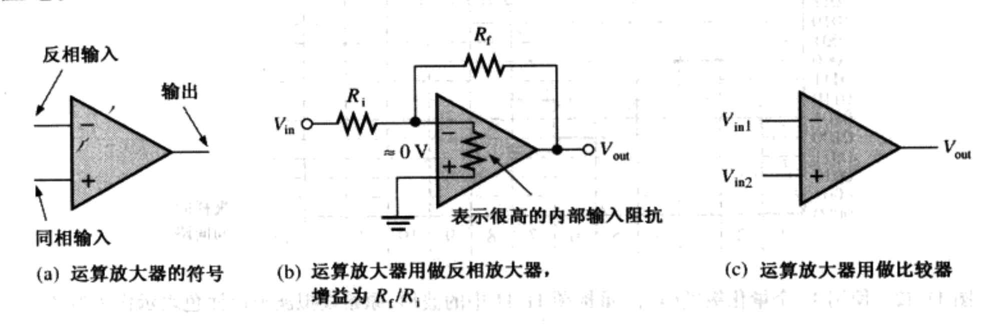
在开始介绍模数转换器(ADC)之前，我们先简单了解一种元件，它对于多数类型的ADC和数模转换器(DAC)都是常用的元件。这种元件就是运算放大器，简称op-amp，即运算放大器的英文缩写。

运算放大器是一种线性放大器，它有两个输入(反相和同相)和一个输出。运算放大器有一个很高的电压增益和很大的输入阻抗，以及很低的输出阻抗。运算放大器的符号如

图 11.13(a)所示。如果用做反相放大器,那么运算放大器的连接如图 11.13(b)所示。根据式(11.2),反馈电阻 R_f 和输入电阻 R_i 控制电压增益,其中 V_{out}/V_{in}是闭环电压增益(闭环是指由电阻提供的从输出到输入的反馈)。负号表示反相。

$$\frac{V_{out}}{V_{in}} = -\frac{R_f}{R_i} \tag{11.2}$$

在反相放大器的连接中,运算放大器的反相输入接近地电位(0 V),因为反馈和极高的开环增益使得两个输入之间的差分电压非常小。由于同相输入接地,反相输入接近 0 V,所以称为虚地。



(a) 运算放大器的符号 (b) 运算放大器用做反相放大器,增益为 R_f/R_i (c) 运算放大器用做比较器

图 11.13 运算放大器

如果把运算放大器用做比较器,如图 11.13(c)所示,两个电压加到输入。当两个输入电压有很小的差别时,运算放大器的输出就进入两种饱和状态的一种,即高电平或低电平,这取决于哪一个输入电压大。

11.3.2 快速(同时)模数转换器

快速方法利用比较器在参考电压和模拟输入电压之间进行比较。如果给定比较器的输入电压超过参考电压,就产生高电平。图 11.14 给出了一个 3 位转换器,使用了 7 个比较器的电路;对于全部为 0 的情况不需要比较器。这种类型的 4 位转换器需要 15 个比较器。一般情况下,对于 n 位二进制码的转换,需要 2^n-1 个比较器。ADC 所用的位数就是它的分辨率。对于适当位数的二进制数需要大量的比较器,这是快速 ADC 的缺点之一。由于它的一个高“通过量”,快速 ADC 的主要优点是提供了一个快速转换时间,以每秒采样次数(sps)来度量。

每个比较器的参考电压都由电阻分压电路设定。每个比较器的输出都连接到优先编码器的一个输入。编码器由 EN 输入上的一个脉冲使能,3 位码表示出现在编码器输出上的输入值。二进制码由最高级输入的高电平决定。

二进制码的序列表示 ADC 的输入,转换的精度由使能脉冲的频率和二进制编码的位数确定。对于输入信号的每个采样电平都需要一个使能脉冲。

例 11.1 请针对图 11.15 中的输入信号和编码器使能脉冲,确定图 11.14 中的 3 位快速 ADC 的二进制编码输出。本例中 $V_{REF}=8$ V。

解:结果得到的数字输出序列如下列出,并给出了和使能脉冲相关的图 11.16 的波形图:

100, 110, 111, 110, 100, 010, 000, 001, 011, 101, 110, 111

相关问题：如果把图 11.15 中的使能脉冲的频率减半，确定由 6 个脉冲得到的数字输出序列所表示的二进制数字。是否丢失了信息？

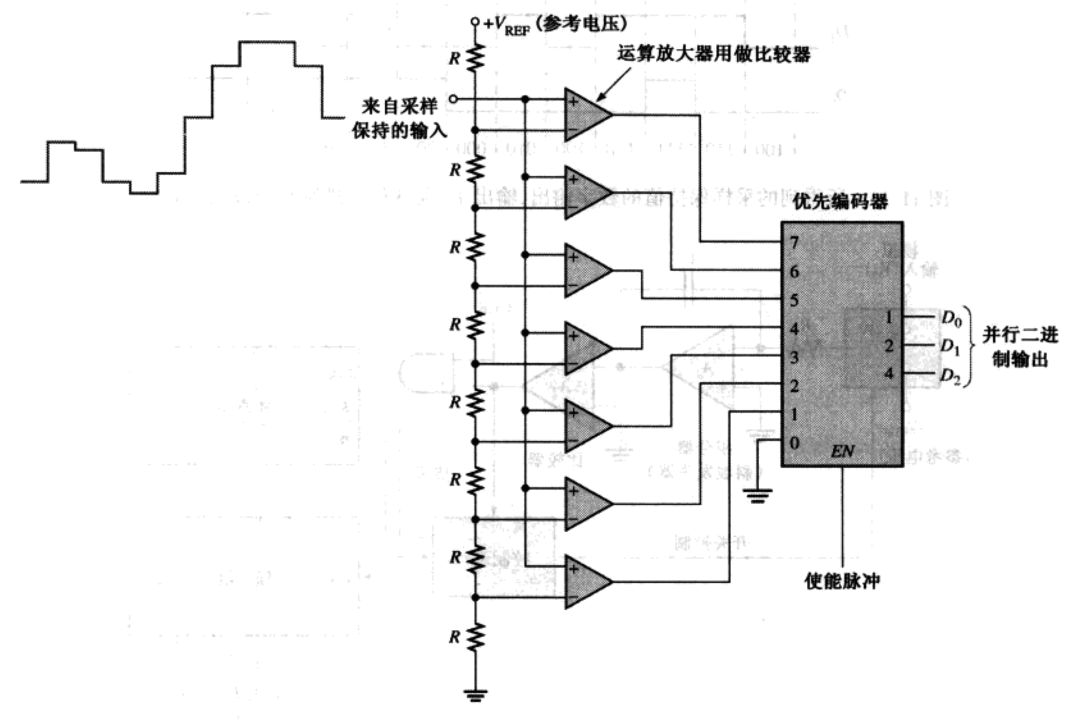


图 11.14 3 位快速 ADC

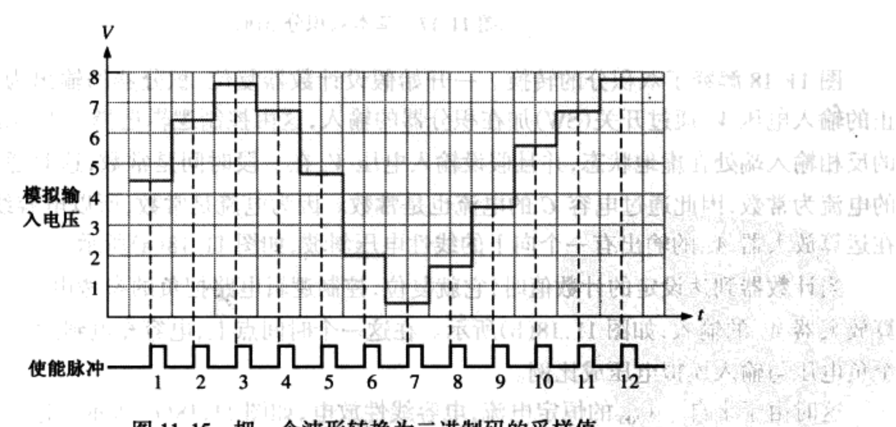


图 11.15 把一个波形转换为二进制码的采样值

11.3.3 双积分模数转换器

双积分 ADC 在数字电压表和其他类型的测量仪器中很常见。一个斜坡发生器（积分器）用来产生双积分特性。双积分 ADC 的框图如图 11.17 所示。

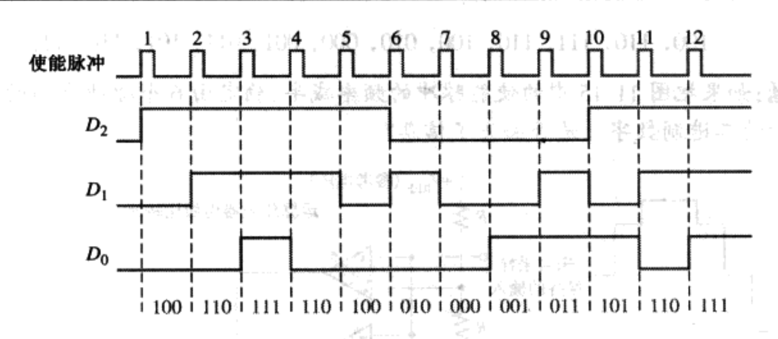


图 11.16 所得到的采样保持值的数字输出,输出 D_0 是 3 位二进制码的最低有效位

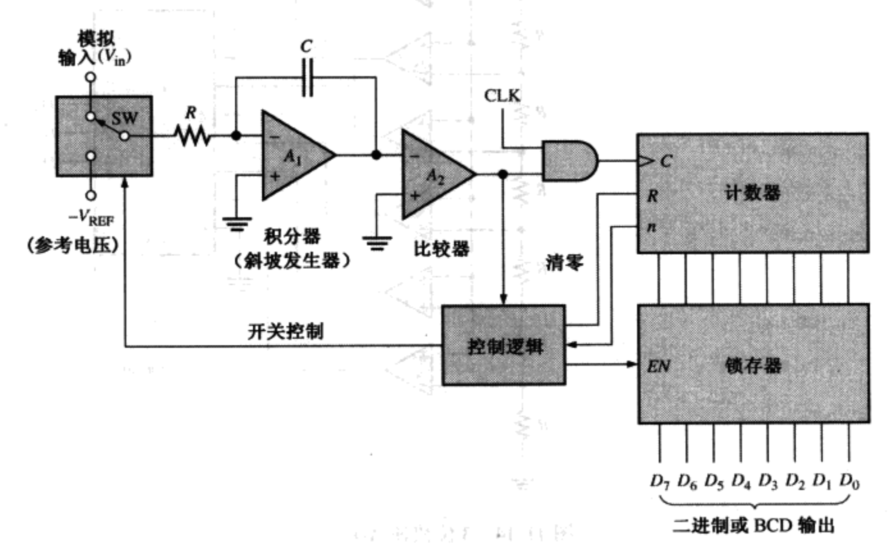


图 11.17 基本双积分 ADC

图 11.18 解释了双积分的转换。一开始假设计数器复位、积分器的输出为零。这时假设正的输入电压 V_{in}通过开关(SW)加在积分器的输入,这由控制逻辑选择。因为运算放大器 A_1 的反相输入端处在虚地状态,并且假设输入电压 V_{in}在一段时间是常数,这时通过输入电阻 R 的电流为常数,因此通过电容 C 的电流也是常数。因为电流是常数,所以电容线性充电,结果在运算放大器 A_1 的输出有一个向下的线性电压斜坡,如图 11.18(a)所示。

当计数器到达设定的计数值时,它就复位,控制逻辑电路把负的参考电压 $-V_{REF}$切换到运算放大器 A_1 的输入,如图 11.18(b)所示。在这一个时间点上,电容充电到达负电压($-V$),这个负电压与输入模拟电压成比例。

这时由于来自 $-V_{REF}$的恒定电流,电容线性放电,如图 11.18(c)所示。这个线性放电过程在运算放大器 A_1 的输出产生一个正向斜坡,斜坡从负电压($-V$)开始,有一个恒定的斜率,这个斜率与前面的充电电压无关。随着电容的放电,计数器从复位状态开始计数。由于放电的速率(斜率)是不变的,电容放电到零所用的时间取决于最初的电压值 $-V$(正比于 V_{in})。当积分器(A_1)的输出电压到达零时,比较器(A_2)就切换到低电平状态,并且关闭计数器的时钟。

这时二进制计数值被锁存,从而完成了一次转换周期。因为电容放电所用的时间仅仅和 $-V$ 有关,所以二进制计数值与 V_{in} 成比例,而计数器记录了这个时间间隔。

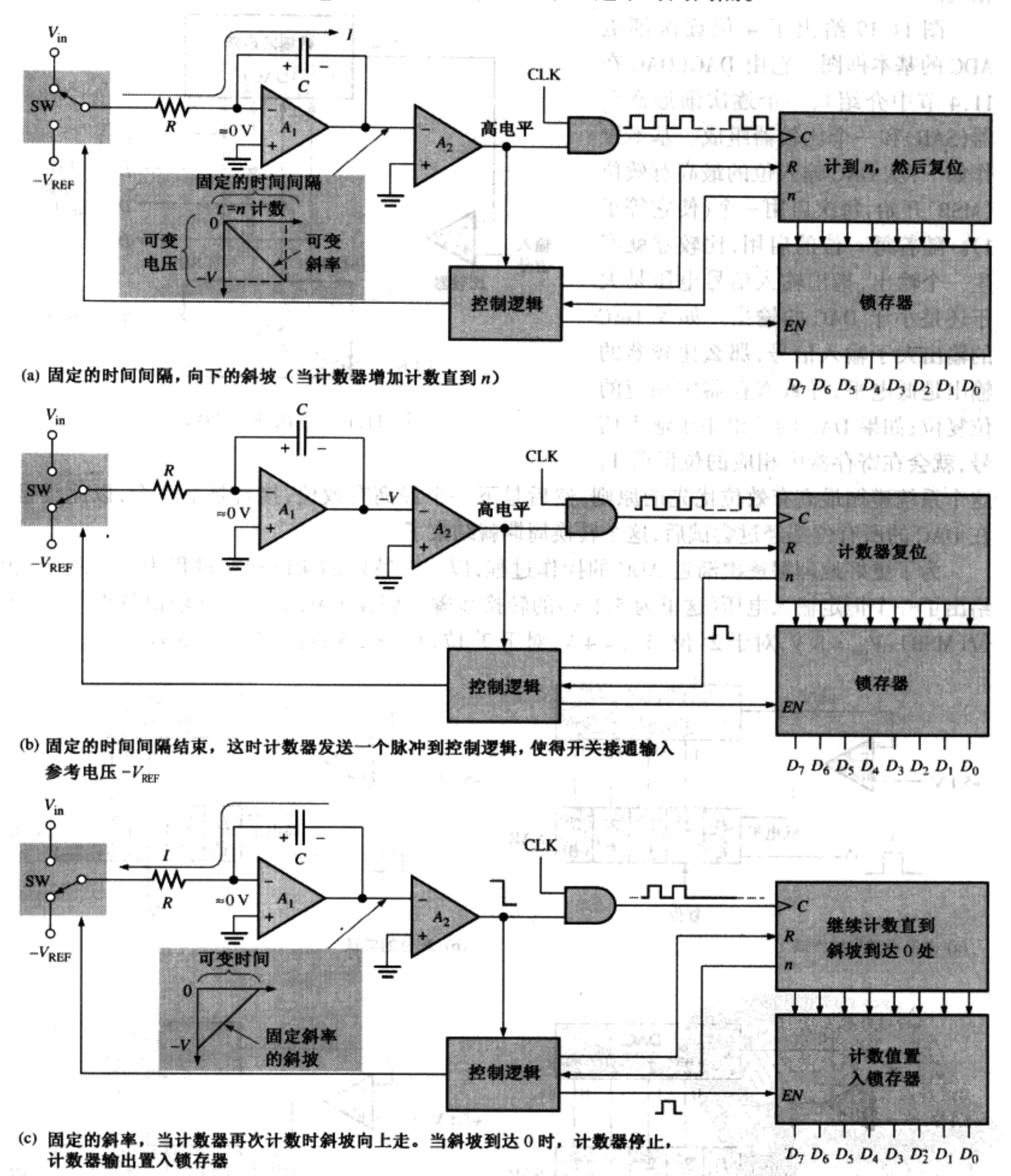


图 11.18 双积分转换的解释

11.3.4 逐次渐近型模数转换器

模数转换最广泛的使用方法之一是逐次渐近。这种方法的转换时间比双积分转换时间

少,但比快速方法要多。这种方法有一个固定的转换时间,对于任何模拟输入值,转换的时间相同。

图 11.19 给出了 4 位逐次渐近 ADC 的基本框图。它由 DAC(DAC 在 11.4 节中介绍)、一个逐次渐近寄存器(SAR)和一个比较器组成。基本操作如下:从 DAC 输入位的最高有效位(MSB)开始,每次启用一个(使它等于 1)。随着每一位的启用,比较器就产生一个输出,指出输入信号电压是大于还是小于 DAC 的输出。如果 DAC 的输出大于输入信号,那么比较器的输出是低电平,导致寄存器中相应的位复位;如果 DAC 的输出小于输入信号,就会在寄存器中相应的位保留 1。这个系统遵循最高有效位优先的原则,然后是下一个最高有效位,接着是下一个,以此类推。在 DAC 的所有位都经过尝试后,这个转换周期就结束了。

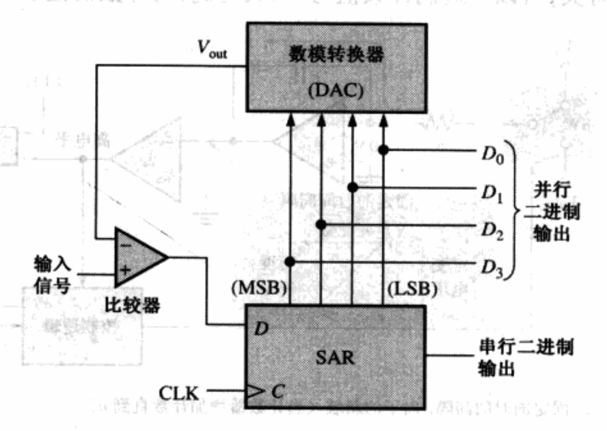


图 11.19 逐次渐近 ADC

为了更好地理解逐次渐近 ADC 的操作过程,以一个特定的 4 位转换过程为例。图 11.20 给出了一个恒定输入电压(这里为 5.1 V)的转换步骤。假设 DAC 具有下列输出特性:对于 2^3 位(MSB),$V_{out}=8$ V;对于 2^2 位,$V_{out}=4$ V;对于 2^1 位,$V_{out}=2$ V;对于 2^0 位(LSB),$V_{out}=1$ V。

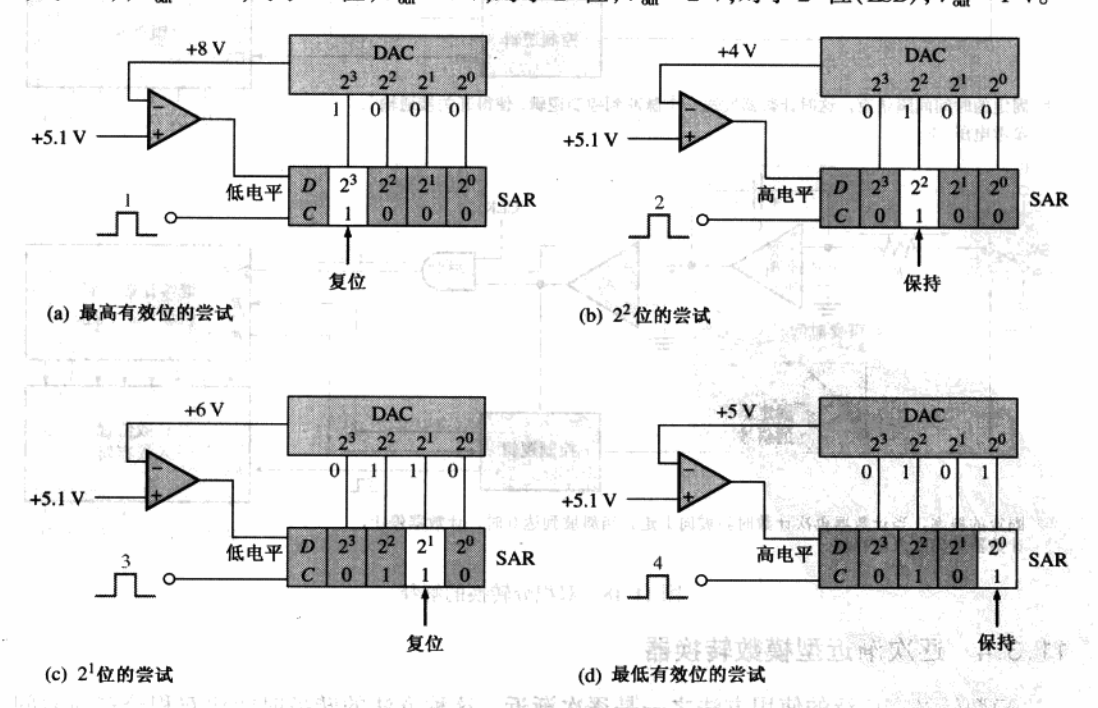


图 11.20 逐次渐近转换过程的示意图

图 11.20(a)给出了转换周期的第一步,最高有效位 MSB = 1。DAC 的输出是 8 V。因为这个输出电压大于 5.1 V,所以比较器的输出是低电平,使得逐次渐近寄存器(SAR)中的最高有效位复位(0)。

图 11.20(b)给出了转换周期的第二步,2^2 位等于 1。DAC 的输出是 4 V。这个输出电压小于 5.1 V,所以比较器的输出转向高电平,使得逐次渐近寄存器(SAR)中的这位保持为 1。

图 11.20(c)给出了转换周期的第三步,2^1 位等于 1。DAC 的输出是 6 V。这个输出电压大于 5.1 V,所以比较器的输出转向低电平,使得逐次渐近寄存器(SAR)中的这位复位(0)。

图 11.20(d)给出了转换周期的第四步也是最后一步,2^0 位等于 1。因为在 2^2 位输入和 2^0 位输入都是 1;4 V + 1 V = 5 V,所以 DAC 的输出是 5 V。

这四位都经过了尝试,所以转换过程结束。此时,寄存器中的二进制编码为 0101,这个值接近输入 5.1 V 的二进制值。增加位数将得到更精确的结果。这时又开始另一个转换周期,这个基本过程又重复。在转换周期的开始,逐次渐近寄存器(SAR)清零。

ADC0804 模数转换器

ADC0804 是逐次渐近 ADC 的一个例子。它的框图如图 11.21 所示。这个芯片的电源为 +5 V,8 位分辨率,转换时间为 100 μs。另外,芯片有一个自带的时钟发生器。三态数据输出,因此可以和微处理器的总线系统接口。

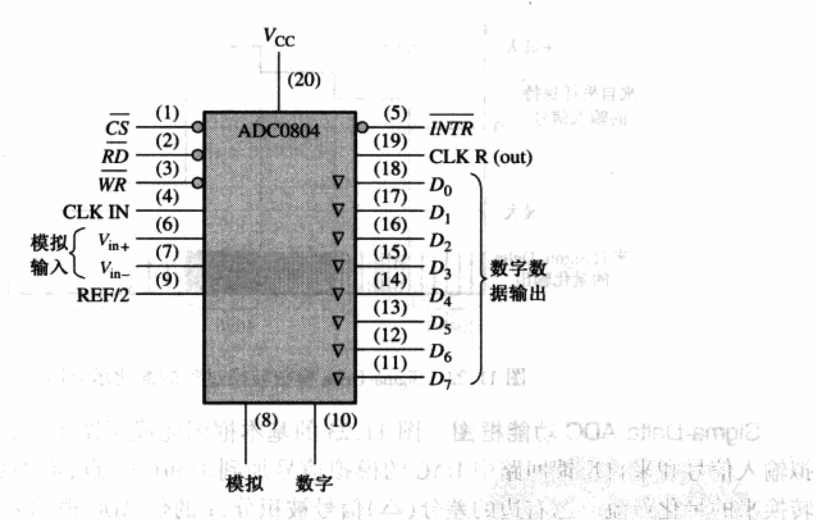


图 11.21 ADC0804 模数转换器

这种芯片的基本操作如下:ADC0804 包含一个 256 电阻 DAC 网络的等效电路。逐次渐近逻辑对网络电路进行排序,以便使模拟输入电压差($V_{in+} - V_{in-}$)与电阻网络的输出匹配。首先测试最高有效位。八次比较后(64 个时钟周期),8 位二进制码就传送到输出锁存器中,中断($\overline{INTR}$)输出变为低电平。通过把$\overline{INTR}$输出连接到写($\overline{WR}$)输入,并使转换开始($\overline{CS}$)保持为低电平,芯片可以工作在自由运行模式。为了确保在所有条件下都可启动,在通电周期时,需要$\overline{WR}$输入为低电平。任何时刻使$\overline{CS}$为低电平随后都会中断转换过程。

当$\overline{WR}$输入变为低电平时，内部逐次渐近寄存器(SAR)和8位移位寄存器都会复位。只要$\overline{CS}$和$\overline{WR}$保持低电平，ADC就保持在复位状态。在$\overline{CS}$或$\overline{WR}$从低电平变到高电平后的一到八个时钟周期，模数转换开始。

当$\overline{CS}$和$\overline{RD}$输入都是低电平时，三态输出锁存器使能，并且输出码加到$D_0 \sim D_7$线上。当$\overline{CS}$或$\overline{RD}$输入中的一个回到高电平，就会禁用$D_0 \sim D_7$输出。

11.3.5 Sigma-Delta 模数转换器

Sigma-Delta(求和 - 增量)是在模数转换过程中广泛使用的方法，特别是在使用音频信号的通信领域。这种方法基于增量(Delta)调制，即对两次连续采样间的差值(增加或减少)进行量化；另一种ADC方法是基于采样的绝对值。增量调制是一种1的位量化方法。

增量调制器的输出是一个单一的位数据流，其中相关的1和0的数目表示输入信号的电平或幅度。1的数目超过给定时钟周期数，超出的数目确立为在这个时间间隔期间的振幅。1的数目的最大值对应于输入正电压的最大值。1的数目的一半对应于输入电压为零。没有1时(全0)对应于负电压的最大值。图11.22以简化方式说明了这点。例如，假设输入信号为正的最大值的间隔期间出现了4096个1。因为零是输入信号动态范围内的中间点，所以当输入信号为零时，间隔期间会出现2048个1。当输入信号为负的最大值时，那么在间隔期间没有1。对于两者之间的信号电平，1的数目与电平值成正比。

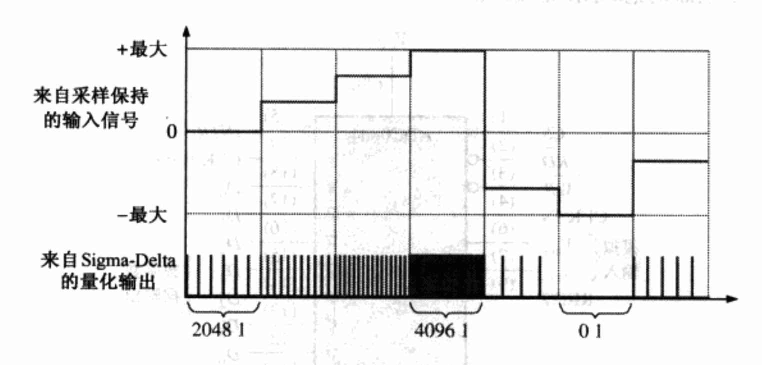


图 11.22 Sigma-Delta 模数转换过程的简化示意图

Sigma-Delta ADC 功能框图 图11.23的基本框图完成了图11.22所示的转换过程。模拟输入信号和来自反馈回路中DAC的模拟信号加到求和(Σ)点，而DAC的模拟信号来自于转换来的量化位流。Σ右边的差分(△)信号被积分，1的位ADC根据差分信号的变化增加或减少1的数目。这样做的目的是为了保持反馈回来的量化信号与进入的模拟信号相等。1的位量化器本质上就是比较器后面跟锁存器。

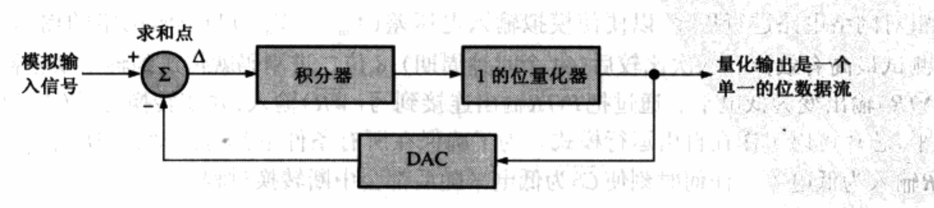


图 11.23 Sigma-Delta ADC 的部分功能框图

为了使用一种特殊方法完成 Sigma-Delta 转换过程，一个单一的位数据流转换成为一系列的二进制编码，如图 11.24 所示。计数器对连续间隔的量化数据流中的 1 进行计数。这时计数器中的码表示每个时间间隔的模拟输入信号的幅度。这些码移位进入锁存器暂时保存。锁存器的输出是一系列的 n 位码，它们完整地表示这个模拟信号。

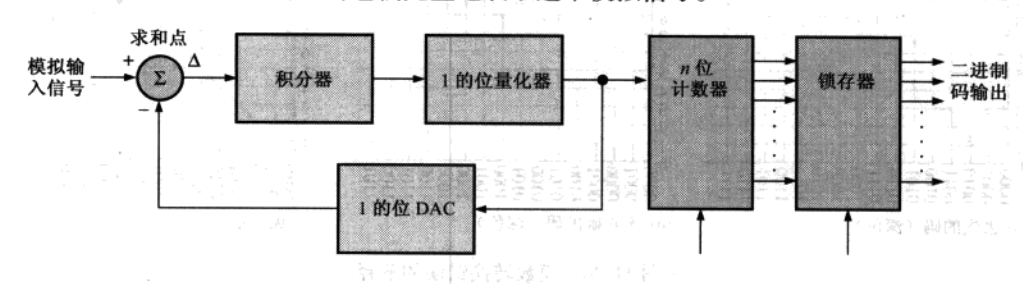


图 11.24　一种类型的 Sigma-Delta ADC

另一种方法是使用数字十进制滤波器，它产生输出信号，而不是计数器和锁存器。这个问题已经超出了本书的讨论范围。

11.3.6　模数转换器的测试

测试 ADC 的一种方法如图 11.25 所示。一个 DAC 用做测试设备的一部分，它把 ADC 的输出转换回模拟形式，用来与测试输入信号进行比较。

测试输入信号以线性斜波的形式施加在 ADC 的输入上。然后把所得到的二进制输出序列加在 DAC 测试单元，并被转换为楼梯状的斜坡。输入和输出的斜坡进行比较，确定是否有偏差。

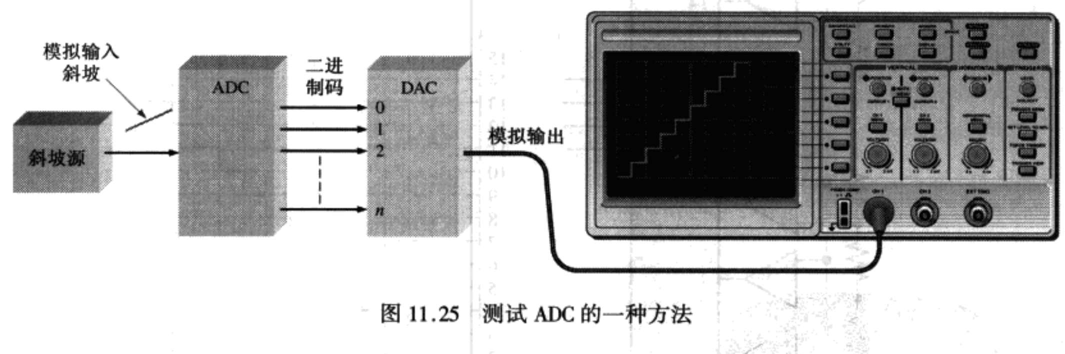


图 11.25　测试 ADC 的一种方法

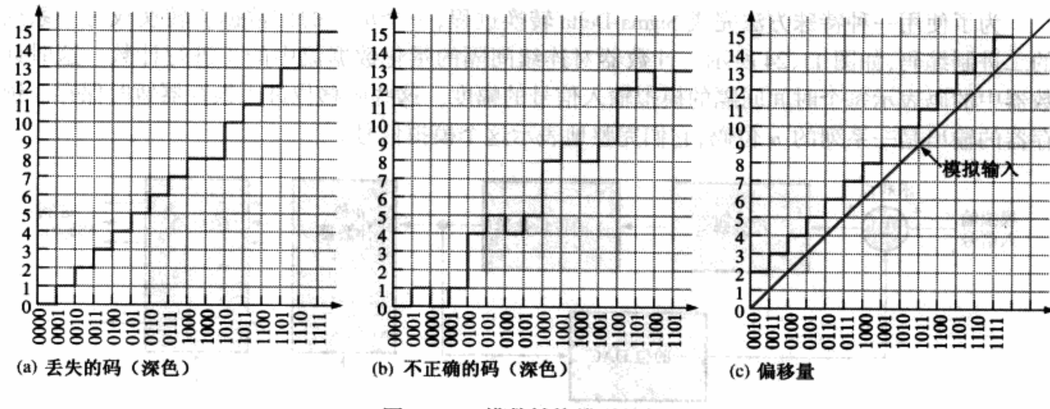
11.3.7　模数转换的错误

再次使用 4 位转换过程来说明这些原则。假定测试输入信号是一个理想的线性斜波。

丢失的码　如图 11.26(a)所示的楼梯状输出，它表示二进制码 1001 没有出现在 ADC 的输出。注意，码值 1000 保持了两个时间间隔，然后输出跳到 1010。

例如，在快速 ADC 中，运算放大器比较器的一次失败操作可能导致丢失码的错误。

不正确的码　如图 11.26(b)所示的楼梯状输出，它表示有几个从 ADC 出来的二进制码字是不正确的。经过分析表明，在这种特殊情况下，2^1 位线停留在低电平(0)状态。

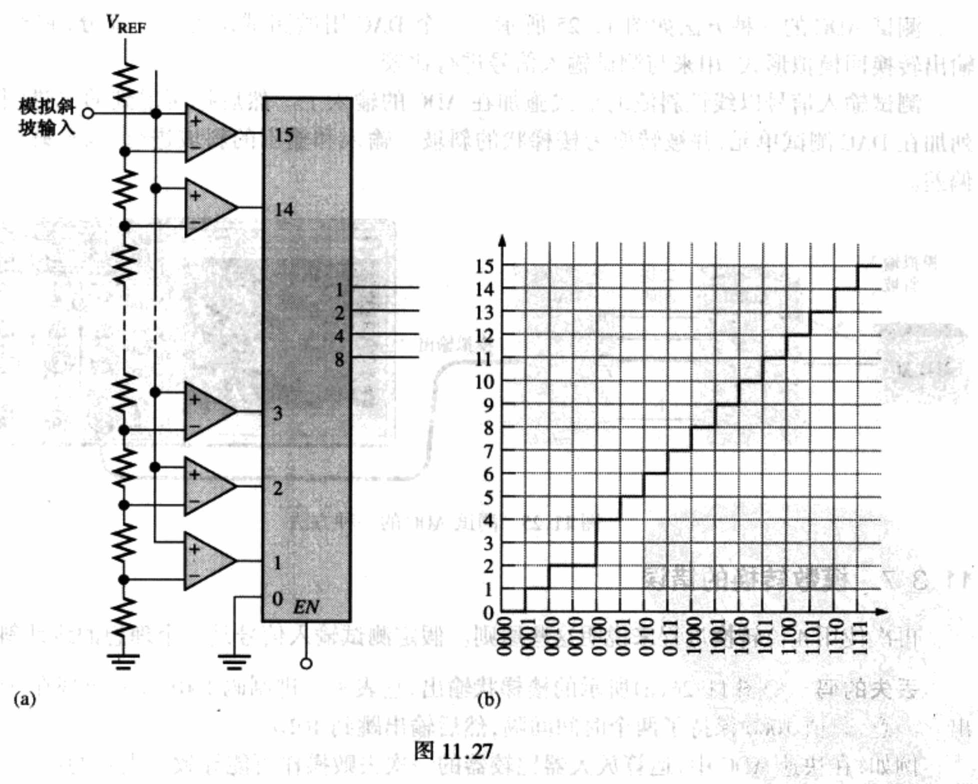


(a) 丢失的码（深色） (b) 不正确的码（深色） (c) 偏移量

图 11.26 模数转换错误的解释

偏移量 偏移量的情况如图 11.26(c)所示。在这种情况下，ADC 理解为模拟输入电压比实际的值大。

例 11.2 如图 11.27(a)所示是一个 4 位快速 ADC。它由一个和图 11.25 相同的设备进行测试。得到的重构模拟输出如图 11.27(b)所示。请找出最有可能出现的错误。



(a) (b)

图 11.27

解：缺少的台阶表明 ADC 的输出丢失二进制码 0011。最大可能是比较器 3 的输出停留在无效状态(低电平)。

相关问题：假定图 11.27(a)中的 ADC 的比较器 8 停留在高电平输出状态，请在如图 11.25 所示的测试设备中重构模拟输出。

11.4 数模转换方法

数模转换是数字处理系统中的重要部分。一旦数字数据经过了 DSP 的处理，就会转换回模拟形式。在本节中，将介绍两种基本类型的数模转换器(DAC)的操作原理，并了解其性能特点。

学完本节以后，应当能够

- 解释二进制权输入 DAC 的操作
- 解释一个 $R/2R$ 阶梯 DAC 的操作
- 讨论 DAC 的分辨率、精度、线性度、单调性和建立时间
- 讨论对 DAC 的非单调性、差分非线性、低或高增益及偏移量误差的测试

11.4.1 二进制权输入数模转换器

数模转换的一种方法是使用电阻网络，网络中的电阻值表示数字码输入位的二进制权值。图 11.28 给出了一个这种类型的 4 位 DAC。每个输入电阻可以有电流，也可以没有电流，这取决于输入电压的电平。如果输入电压为零(二进制 0)，那么电流也为零。如果输入电压为高电平(二进制 1)，那么电流的大小取决于输入电阻的值，因而对于每个输入电阻其电流不同，如图所示。

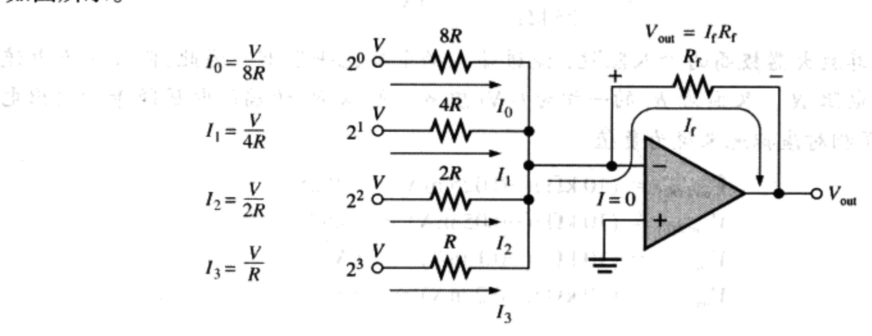


图 11.28 一个二进制权输入的 4 位 DAC

因为运算放大器的反向输入(－)几乎没有电流进入，所以所有输入电流将加在一起，并经过电阻 R_f。因为反向输入为 0 V(虚地)，所以 R_f 两端的电压降等于输出电压，因此 $V_{out}=I_fR_f$。

输入电阻值的选择和相应输入位的二进制权值成反比。电阻的最小值(R)对应于最大的二进制权输入(2^3)。其他的电阻值为 R 的倍数(即 $2R$、$4R$ 和 $8R$)，分别对应于二进制权值 2^2、2^1、2^0。输入电流也和二进制权值成正比。因此，输出电压与二进制权值的和成正比，因为输入电流的和经过 R_f。

这种类型的 DAC 有一定缺点，就是不同电阻值的数目较多，所有输入的电压电平必须完全一样。例如，一个 8 位转换器需要 8 个电阻，并且电阻值的范围以二进制权的步长从 R 变化到 $128R$。电阻的范围需要 255 分之一(小于 0.5%)的允许误差才能精确地转换输入。因此这种类型的 DAC 很难大量生产。

例 11.3 确定图 11.29(a)中 DAC 的输出，波形表示在图 11.29(b) 中加到输入的 4 位数的序列，输 D_0 是最低有效位(LSB)。

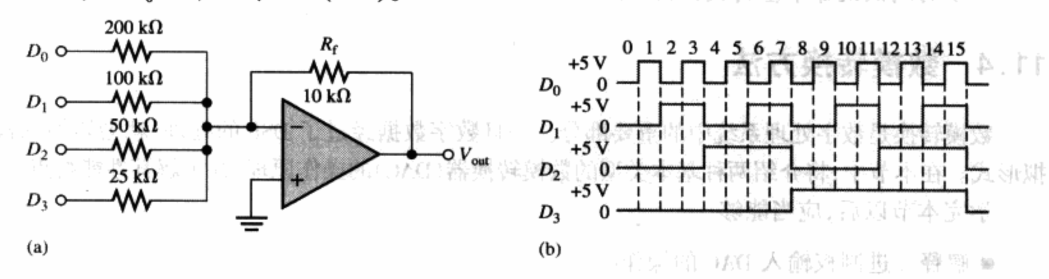


图 11.29

解：首先，确定每个权输入的电流。因为运算放大器的反向输入(－)是 0 V(虚地)，而二进制 1 对应于 +5 V，所以经过任意一个输入电阻的电流是 5 V 除以电阻值。

$$I_0 = \frac{5\text{ V}}{200\text{ k}\Omega} = 0.025\text{ mA}$$

$$I_1 = \frac{5\text{ V}}{100\text{ k}\Omega} = 0.05\text{ mA}$$

$$I_2 = \frac{5\text{ V}}{50\text{ k}\Omega} = 0.1\text{ mA}$$

$$I_3 = \frac{5\text{ V}}{25\text{ k}\Omega} = 0.2\text{ mA}$$

因为反向运算放大器极高的输入阻抗，流进输入的电流几乎没有。因此，假设所有电流都经过反馈电阻 R_f。又因为 R_f 的一端为 0 V(虚地)，所以 R_f 两端的电压降等于输出电压，输出电压相对虚拟地来说为负值。

$$V_{out(D0)} = (10\text{ k}\Omega)(-0.025\text{ mA}) = -0.25\text{ V}$$

$$V_{out(D1)} = (10\text{ k}\Omega)(-0.05\text{ mA}) = -0.5\text{ V}$$

$$V_{out(D2)} = (10\text{ k}\Omega)(-0.1\text{ mA}) = -1\text{ V}$$

$$V_{out(D3)} = (10\text{ k}\Omega)(-0.2\text{ mA}) = -2\text{ V}$$

从图 11.29(b)可知，第一个二进制输入码是 0000，它产生了 0 V 的输出电压。下一个输入码是 0001，它产生了 －0.25 V 的输出电压。因此，输出电压为 －0.25 V。接下来的码是 0010，它产生了 －0.5 V 的输出电压。下一个码是 0011，它产生的输出电压是 －0.25 V＋(－0.5 V)＝－0.75 V。每个连续的二进制码都将使输出电压增大 －0.25 V，因此对于这种特殊的直线二进制输入序列，输出是从 0 V 到 －3.75 V 的阶梯形波，步长为 －0.25 V。如图 11.30所示。

相关问题：将图 11.29 中到 DAC 的输入波形翻转(D_3 到 D_0，D_2 到 D_1，D_1 到 D_2，D_0 到 D_3)，然后确定其输出。

11.4.2 *R*/2*R* 阶梯型数模转换器

数模转换的另一种方法是 *R*/2*R* 阶梯型，如图 11.31 所示的四位 DAC。这种方法解决了二进制权输入 DAC 的一个问题，它只需要两个电阻值。

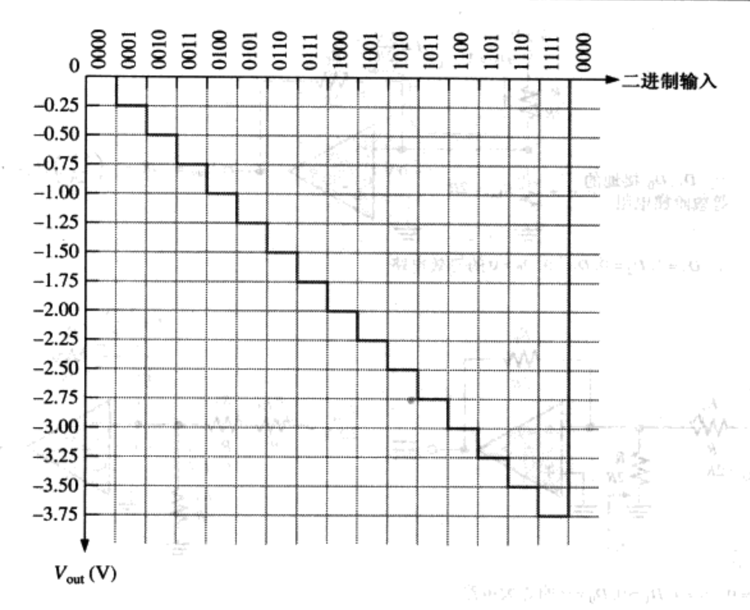


图 11.30　图 11.29 中 DAC 的输出

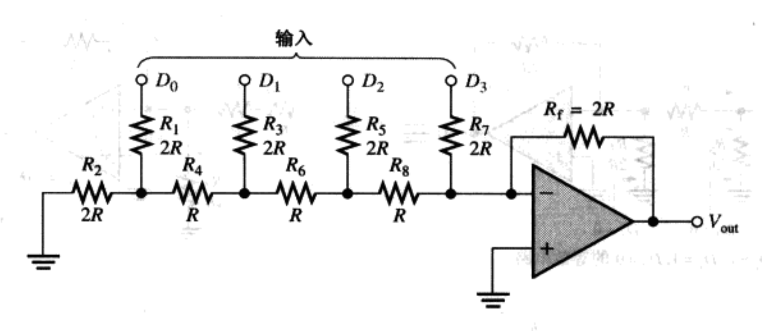
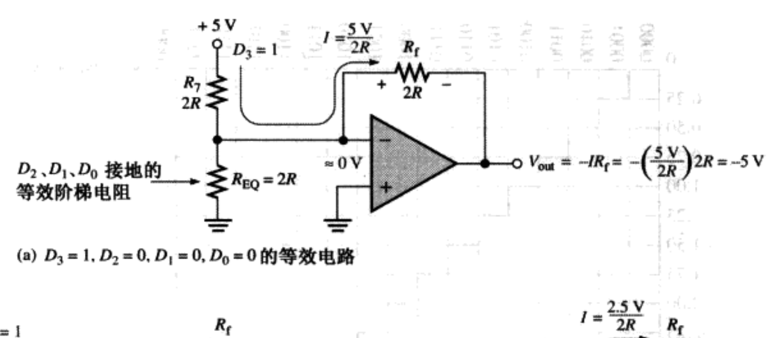


图 11.31　$R/2R$ 阶梯型 DAC

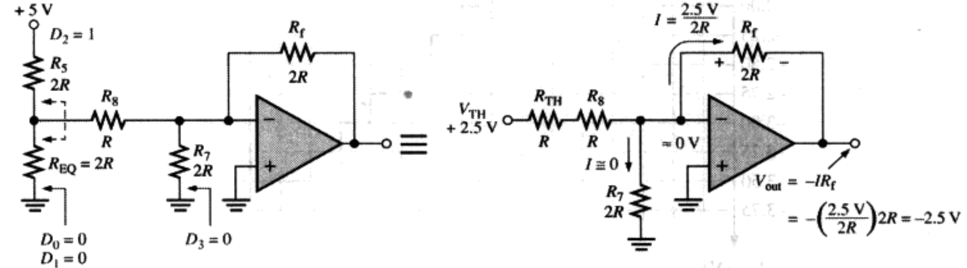
假定开始时输入 D_3 为高电平(+5 V),其他都为低电平(接地,0 V)。这种情况表示二进制数 1000。电路分析指出这个情况能够简化为如图 11.32(a)所示的等价电路。本质上,没有电流经过 $2R$ 等价电阻(除了 R_7),因为反向输入是虚地。因此,通过 R_7 的电流($I = 5\ V/2R$)也通过 R_f,而输出电压为 -5 V。由于负反馈,运算放大器将保持反向输入(-)接近零电压(≈ 0 V)。因此,所有的电流将通过,而不是进入反向输入。

图 11.32(b)给出了当 D_2 输入等于 +5 V 而其他输入接地时的等效电路。这个情况表示 0100。根据戴维宁(Thevenin)定理,从 R_8 向左看,得到 2.5 V 和与之串联的电阻 R,如图所示。结果得到通过 R_f 的电流 $I = 2.5\ V/2R$,其输出电压是 -2.5 V。记住,运算放大器的反向输入没有电流进入,也没有电流通过等效的接地电阻,因为虚地的原因,电阻两端的电压为 0 V。

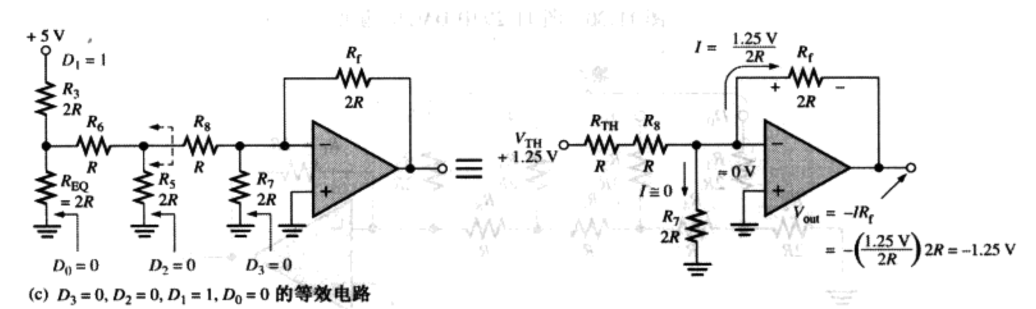
图 11.32(c)给出了当 D_1 输入等于 +5 V 而其他输入接地时的等效电路。这种情况表示 0010。也是根据戴维宁定理,从 R_8 向左看,得到 1.25 V 和与之串联的电阻 R,如图所示。结果得到通过 R_f 的电流 $I = 1.25\ V/2R$,其输出电压是 -1.25 V。



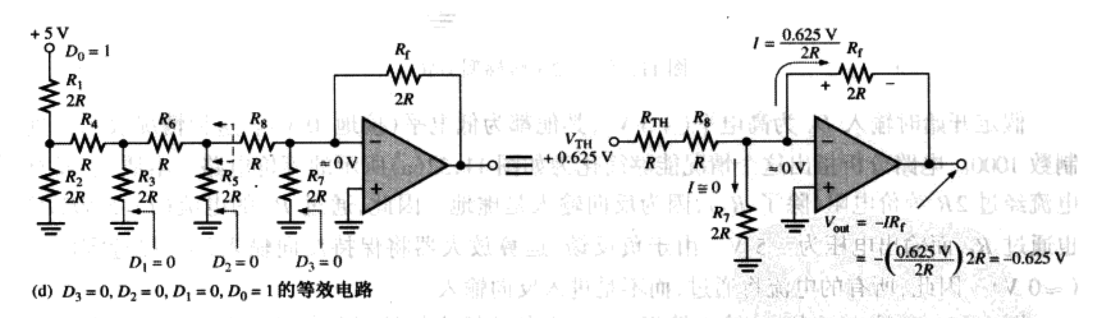
(a) $D_3 = 1, D_2 = 0, D_1 = 0, D_0 = 0$ 的等效电路



(b) $D_3 = 0, D_2 = 1, D_1 = 0, D_0 = 0$ 的等效电路



(c) $D_3 = 0, D_2 = 0, D_1 = 1, D_0 = 0$ 的等效电路



(d) $D_3 = 0, D_2 = 0, D_1 = 0, D_0 = 1$ 的等效电路

图 11.32 *R*/2*R* 阶梯型 DAC 的分析

图 11.32(d)给出了当 D_0 输入等于 +5 V 而其他输入接地时的等效电路。这个情况表示 0001。从 R_8 向左看的戴维宁电路给出等效电压 0.625 V 和与之串联的电阻 R，如图所示。结果得到通过 R_f 的电流 $I = 0.625\ V/2R$，其输出电压是 −0.625 V。

注意，每个连续的较低权输入产生减半的输出电压，这样，输出电压就与输入位的二进制权值成正比。

11.4.3　数模转换器的性能特点

DAC 的性能特点包括分辨率、精度、线性度、单调性及建立时间，下面对每种特性进行讨论：

- **分辨率**：DAC 的分辨率就是输入中离散的增量数的倒数。当然，这是依赖于输入位的数目。例如，一个 4 位 DAC 的分辨率是 $1/(2^4-1)$（十五分之一）。表示为百分数，就是 $(1/15)\cdot 100=6.67\%$。离散阶梯数的总数等于 2^n-1，其中 n 是位的数目。分辨率还可表示为转换的位的数目。
- **精度**：精度是 DAC 实际输出和期望输出的比较而得到的。表示为满量程，或最大输出电压的百分数。例如，一个转换器的满量程输出为 10 V，精度为 ±0.1%，那么任何输出电压的最大误差为(10 V)(0.001) = 10 mV。理想情况下，精度应当不低于最低有效位的 ±1/2。对于一个 8 位转换器，最低有效位是满量程的 0.39 %。精度大约为 ±0.2%。
- **线性度**：线性误差是偏离 DAC 理想直线输出的误差。一种特殊的情况就是偏移误差，它是在输入位全部为零时输出的电压值。
- **单调性**：如果一个 DAC 顺序经历输入位的全部范围，没有获得任何相反的增量，那么此 DAC 是单调的。
- **建立时间**：建立时间通常定义为，从输入码发生变化，到 DAC 在最终值的 ±1/2 最低有效位以内稳定下来所花费的时间。

例 11.4　确定下列情况的分辨率，表示为百分数：

(a)8 位 DAC　　　　(b)12 位 DAC

解：

(a)对于 8 位转换器，

$$\frac{1}{2^8-1}\times 100=\frac{1}{255}\times 100=\mathbf{0.392\%}$$

(b)对于 12 位转换器，

$$\frac{1}{2^{12}-1}\times 100=\frac{1}{4095}\times 100=\mathbf{0.0244\%}$$

相关问题：计算 16 位 DAC 的分辨率。

11.4.4　测试模数转换器

DAC 测试的概念如图 11.33 所示。在这个基本方法中，在输入加一个二进制码序列，观察得到的输出。二进制码序列扩展到满量程，即从 0 到 2^n-1 升序排列的全部值，其中 n 是位的数目。

理想的输出是如图所示的直线型阶梯。随着二进制码的位数增加，分辨率也提高。也就是说，随着离散增量的数量增加，输出更接近于直线线性斜坡。

11.4.5　数模转换错误

图 11.34 给出了几种检测到的数模转换错误，为了举例说明将使用 4 位转换。4 位转换将产生 15 步离散的增量。图中的每个图形都包含一条理想的阶梯状斜坡，作为错误输出的比较。

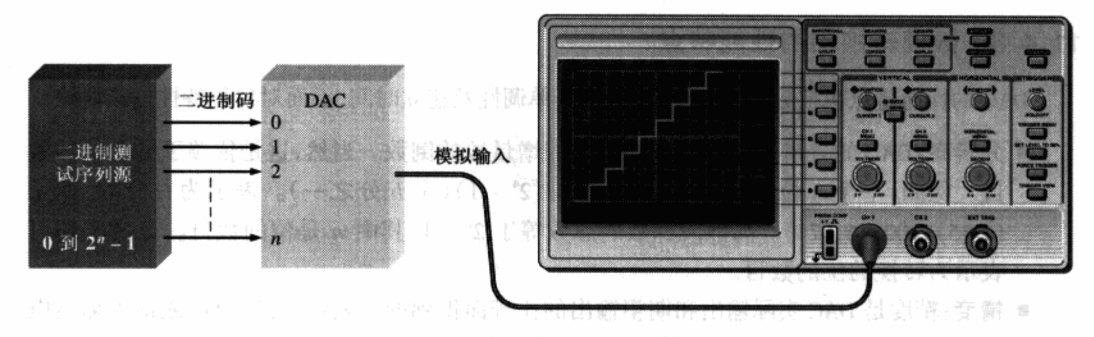


图 11.33 DAC 的基本测试设备

非单调性 图 11.34(a)中颠倒的增量表示了非单调性,这是非线性的一种。在这种特殊的情况下,出现错误是因为二进制编码中的 2^1 位被解释为常数 0。也就是说,一个短路引起输入线路停留在低电平。

模拟输出 理想 二进制输入

(a) 非单调性(粗线)

模拟输出 二进制输入

(b) 非线性偏差(粗线)

模拟输出 高增益 低增益 二进制输入

(c) 高增益和低增益(粗线)

模拟输出 二进制输入

(d) 偏移误差(粗线)

图 11.34 几种数模转换错误的图例

非线性偏差　图 11.34(b)给出了非线性偏差的情况，对于某些输入码，增量的幅度小于它应该有的大小。这种特殊的输出可能是由于 2^2 位的权值不够大，也可能是由于不正确的输入电阻。如果一个特殊的二进制权值大于它应有的值，那么也可以看到增量的幅度大于它应有的值。

低或高增益　由低或高增益引起的输出错误如图 11.34(c)所示。在低增益的情况下，所有增量的幅度都小于理想值。在高增益的情况下，所有增量的幅度都大于理想值。这种情形可能是由运算放大器电路中不正确的反馈电阻引起的。

偏移误差　偏移误差如图 11.34(d)所示。注意，当二进制输入为 0000 时，输出电压不是零，而且这个偏移量的值对于转换中的所有增量都是相同的。不正确的运算放大器可能导致了这种情况。

例 11.5　图 11.35 中为一个直线形 4 位二进制序列加到输入时观察到的 DAC 输出。请辨别错误类型，并给出查出错误的方法。

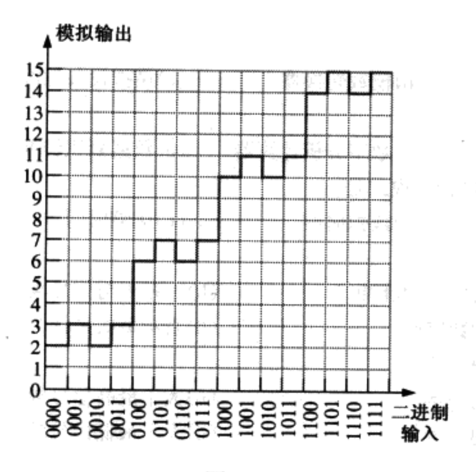


图 11.35

解：这种情况下的 DAC 是非单调的。分析输出发现设备转换成下面的序列，而不是实际施加在输入上的二进制序列。

0010, 0011, 0010, 0011, 0110, 0111, 0110, 0111, 1010, 1011, 1010, 1011, 1110, 1111, 1110, 1111

显然，2^1 位停留在高电平(1)状态。要找到问题，首先应监视设备位输入引脚的状态。如果正在改变状态，那么是 DAC 内部错误，应当将它替换掉。如果外部引脚没有改变状态，并且总是为高电平，那么可能是由于电路板上某个焊接物的跨接造成 $+V$ 和外部短路。

相关问题：当线形的 4 位二进制序列加在输入并且 2^0 位保持为高电平，请确定 DAC 的输出。

11.4.6　重构滤波器

DAC 的输出是一个"阶梯"状的波形，近似于经过 DSP 处理后的原始模拟信号。低通重构

滤波器(有时称为后滤波器)的用途是消除较高的高频成分,以平滑 DAC 的输出。高频成分是"阶梯"的快速转换导致的,图 11.36 进行了大致的描述。

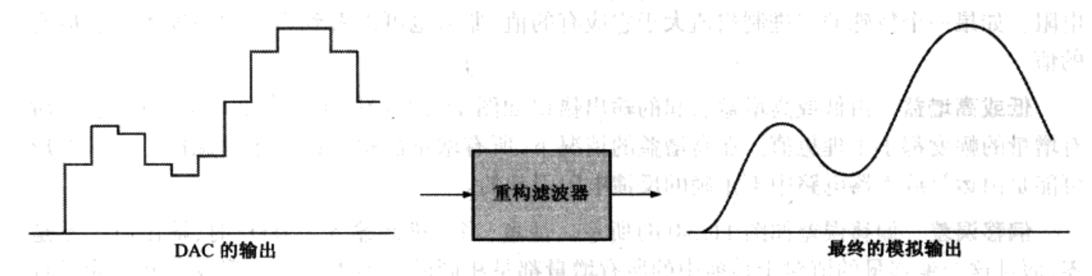


图 11.36 重构滤波器平滑 DAC 的输出

自测题 (答案在本章的结尾)

1. ADC 是
   (a)字母数据编码器 (b)模数转换器 (c)模拟设备载波器 (d)模数比较器
2. DAC 是
   (a)数模计算机 (b)数字分析计算器 (c)数据累积转换器 (d)数模转换器
3. 数字信号处理系统通常的操作模式是
   (a)实时 (b)虚拟时间 (c)压缩时间 (d)计算机时间
4. 模拟信号的采样产生
   (a)一系列与信号幅度成正比的脉冲 (b)一系列与信号频率成正比的脉冲
   (c)表示模拟信号幅度的数字码 (d)表示每次采样时间的数字码
5. 根据采样定理,采样频率应当
   (a)小于最高信号频率的一半 (b)大于最高信号频率的两倍
   (c)小于最低信号频率的一半 (d)大于最低信号频率
6. 保持操作出现在
   (a)每次采样前 (b)每次采样期间 (c)模数转换后 (d)紧跟在采样后
7. 量化过程是
   (a)把采样和保持输出转换成二进制码 (b)把采样脉冲转换成一个电平
   (c)把二进制码序列转换成重构的模拟信号 (d)在采样发生前滤除不需要的频率
8. 一般情况下,模拟信号可通过下列哪种方式更准确地重构?
   (a)更多的量化电平 (b)更少的量化电平 (c)更高的采样频率
   (d)更低的采样频率 (e)答案(a)或(c)
9. 快速 ADC 使用
   (a)计数器 (b)运算放大器 (c)积分器
   (d)触发器 (e)答案(a)和(c)
10. 双积分 ADC 使用
    (a)计数器 (b)运算放大器 (c)积分器
    (d)微分器 (e)答案(a)和(c)
11. 求和-增量(sigma-delta)ADC 的输出是
    (a)并行二进制码 (b)多位数据

(c)单一的位数据 (d)差分电压

12. 在二进制权 DAC 中,输入的电阻将
    (a)决定模拟信号的振幅 (b)决定数字输入的权值
    (c)限制功率损耗 (d)阻止从数据源置入

13. 在一个 $R/2R$ DAC 中,有
    (a)4 个电阻值 (b)1 个电阻值
    (c)2 个电阻值 (d)电阻值的数量等于输入的数量

习题

11.1 节 数字信号处理基础

1. 说明模数转换的用途。
2. 在图 11.37 的数字信号处理系统框图中填写恰当的功能名称。

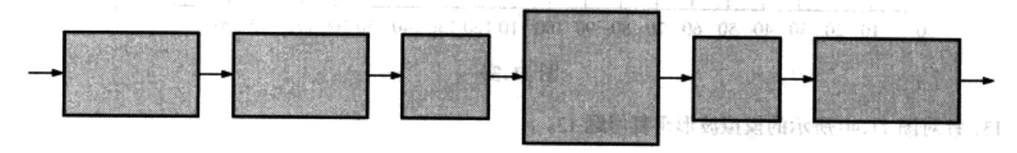


图 11.37

3. 说明数模转换的用途。

11.2 节 模拟信号转换为数字信号

4. 图 11.38 所示的波形加在采样电路上,每 3 ms 采样一次。给出采样电路的输出。假设在输入和输出间是一对一的电压对应关系。

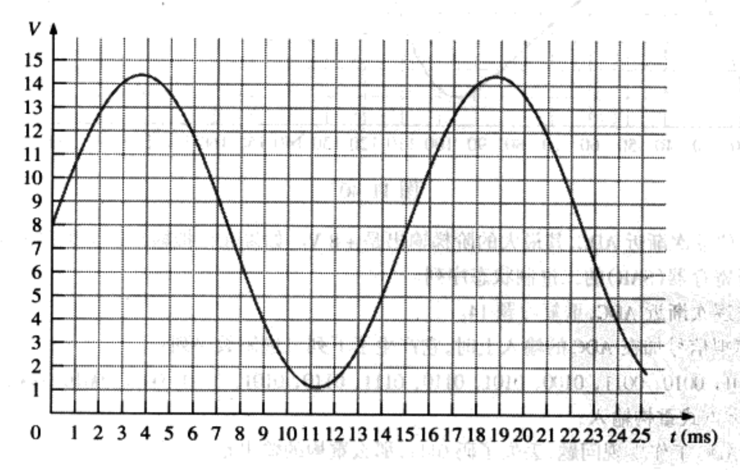


图 11.38

5. 把习题 4 中采样电路的输出加在保持电路上。给出保持电路的输出。
6. 如果将习题 5 中保持电路的输出进行两位量化,那么得到的二进制码序列是怎样的?
7. 请使用 4 位量化重复习题 6。
8. (a)以习题 6 中的 2 位量化序列重构模拟信号。
   (b)以习题 7 中的 4 位量化序列重构模拟信号。

9. 画出由下列二进制数序列表示的模拟函数。

   1111, 1110, 1101, 1100, 1010, 1001, 1000, 0111, 0110, 0101, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1100, 1100, 1011, 1010, 1001

11.3 节　模数转换方法

10. 某个运算放大器组成的反向放大器的输入电压是 10 mV,输出是 2 V。其闭环电压增益是多少?
11. 要使某个反向放大器的闭环电压增益达到 330,如果 $R_i = 1.0\ \text{k}\Omega$,那么反馈电阻应该是多少?
12. 对于如图 11.39 所示的模拟输入信号,确定一个 3 位快速 ADC 的二进制输出码。

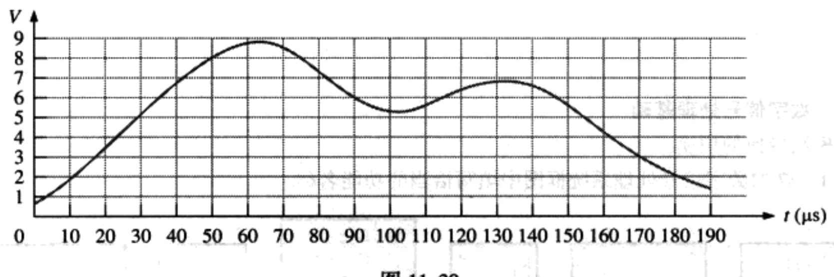


图 11.39

13. 针对图 11.40 所示的模拟波形重复习题 12。

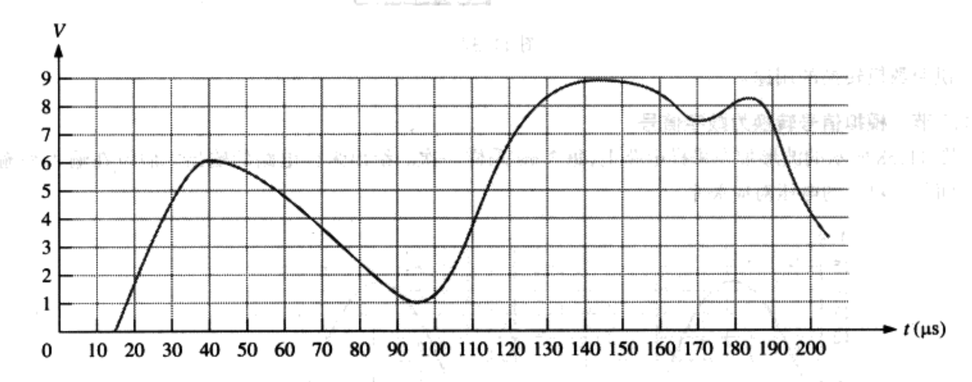


图 11.40

14. 对某个 2 位逐次渐近 ADC,其最大的阶梯输出是 +8 V。如果在模拟输入加上恒定的 +6 V 电压,确定逐次渐近寄存器(SAR)的二进制状态序列。
15. 针对 4 位逐次渐近 ADC,重复习题 14。
16. 当一个模拟信号加在 ADC 的输入上时,它产生了下列二进制数序列:

    0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 0110, 0101, 0100, 0011, 0010, 0001, 0000

    (a)以数字方式重构输入;

    (b)如果 ADC 工作出现问题,丢失了码 0111,那么重构的输出是怎样的?

11.4 节　数模转换方法

17. 在图 11.28 的 4 位 DAC 中,最低权电阻的值是 10 kΩ。其他输入电阻应该是多少?
18. 如果在输入加上如图 11.41(b)所示的 4 位数字序列,确定图 11.41(a)的 DAC 的输出。数据输入的低电压为 0 V,高电压为 +5 V。
19. 针对图 11.42 中的输入,重复习题 18。

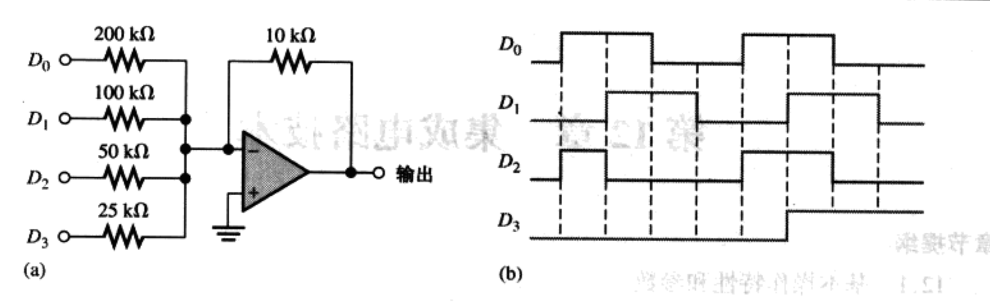


图 11.41

20. 请针对下列每一种 DAC,确定其以百分数表示的分辨率:

 (a)3 位 (b)10 位 (c)18 位

21. 开发一个可产生 8 位二进制的测试序列,参见图 11.33 的测试设备的电路。

22. 一个 4 位 DAC 出现问题,它的最高有效位(MSB)停留在 0 状态。绘制在其输入加上直线二进制序列时的模拟输出。

23. 一个 4 位 DAC 的输入加上了直线形二进制序列,其输出如图 11.43 所示。有什么问题?

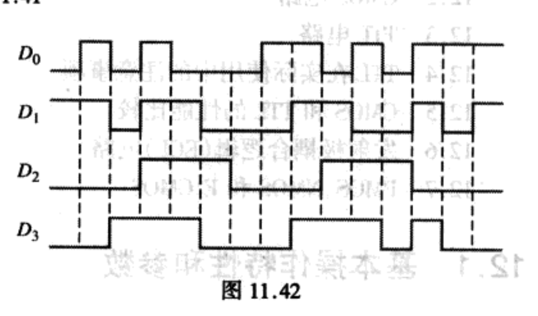


图 11.42

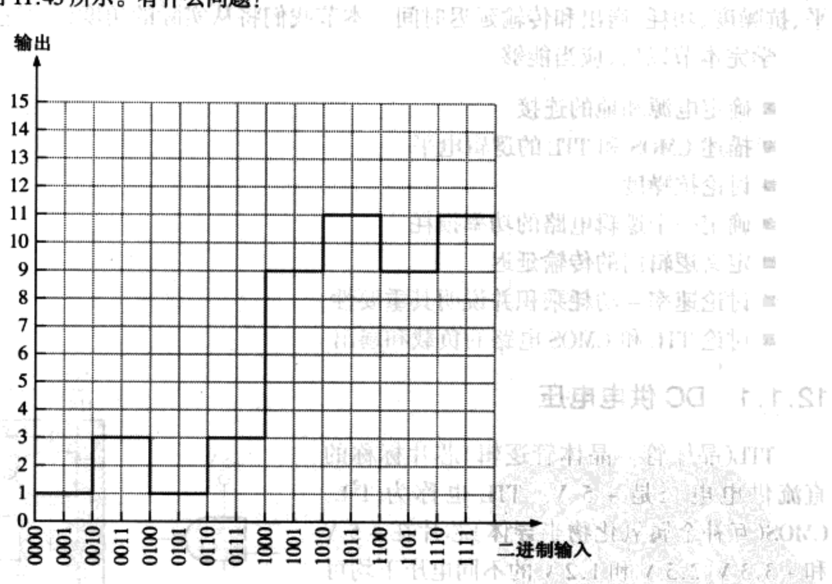


图 11.43

自测题答案

1. (b) 2. (d) 3. (a) 4. (a) 5. (b) 6. (d) 7. (a) 8. (e) 9. (b) 10. (e) 11. (c) 12. (b) 13. (c)

第 12 章　集成电路技术

章节提纲

12.1　基本操作特性和参数

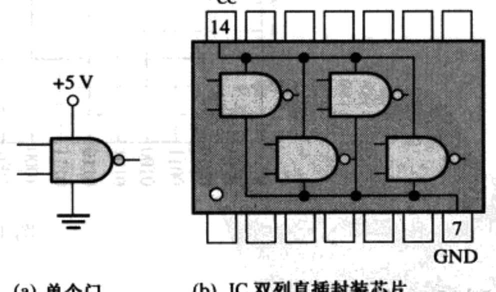
在使用数字 IC 芯片时,不仅应该熟悉其逻辑操作,还应该熟悉一些操作属性,如电压电平、抗噪度、功耗、扇出和传输延迟时间。本节我们将从实际的角度讨论这些属性。

学完本节以后,应当能够

- 确定电源和地的连接
- 描述 CMOS 和 TTL 的逻辑电平
- 讨论抗噪度
- 确定一个逻辑电路的功率损耗
- 定义逻辑门的传输延迟
- 讨论速率 - 功耗乘积并说明其重要性
- 讨论 TTL 和 CMOS 电路的负载和扇出

12.1.1　DC 供电电压

TTL(晶体管一晶体管逻辑)芯片标称的直流供电电压是 + 5 V。TTL 也称为 T^2L。CMOS(互补金属氧化物半导体)芯片在 + 5 V 和 + 3.3 V、2.5 V 和 1.2 V 的不同电压下均可工作。为了简化起见,直流供电电压在逻辑图中被省略了,但它其实是连接在 IC 封装芯片的 V_{CC} 引脚上的,而地则连接 GND 引脚。电压和地在内部被分配给 IC 封装芯片中的所有元件,如图 12.1 所示是一个 12 引脚封装芯片。



(a) 单个门　(b) IC 双列直插封装芯片

图 12.1　V_{CC} 和地的连接和在 IC 封装芯片中的分配情况。为了简化,忽略了其他引脚的连接

12.1.2 CMOS 逻辑电平

第 1 章简单介绍了逻辑电平。有 4 种不同的逻辑电平规范：V_{IL}，V_{IH}，V_{OL}，V_{OH}。对 CMOS 电路来说，输入电压的范围(V_{IL})可以用一个有效低电平(逻辑 0)表示，这个有效低电平对于 5 V 逻辑是从 0 V 到 1.5 V，对于 3.3 V 逻辑是从 0 V 到 0.8 V。表示有效高电平(逻辑 1)的输入电压的范围(V_{IH})对于 5 V 逻辑是 3.5 V 到 5 V，对于 3.3 V 逻辑是 2 V 到 3.3 V，如图 12.2 所示。对于 5 V 逻辑，1.5 V 到 3.5 V，以及对于 3.3 V 逻辑，0.8 V 到 2 V 的电压值范围是不可预测性能的区域，在这些区域的电压值是不允许出现的。当输入电压在这些区域中时，逻辑电路也许将其解释为高电平或低电平。因此，当输入电压出现在这些不允许的区域中时，CMOS 门电路的操作是不可靠的。

对于 5 V 和 3.3 V 逻辑电平，CMOS 输出电压的范围(V_{OL}和 V_{OH})如图 12.2 所示。注意，输出电压的最小高电平 $V_{OH(min)}$ 比输入电压的最小高电平 $V_{IH(min)}$ 要大。同样，还要注意输出电压的最大低电平值 $V_{OL(max)}$ 比输入电压的最大低电平 $V_{IL(max)}$ 要小。

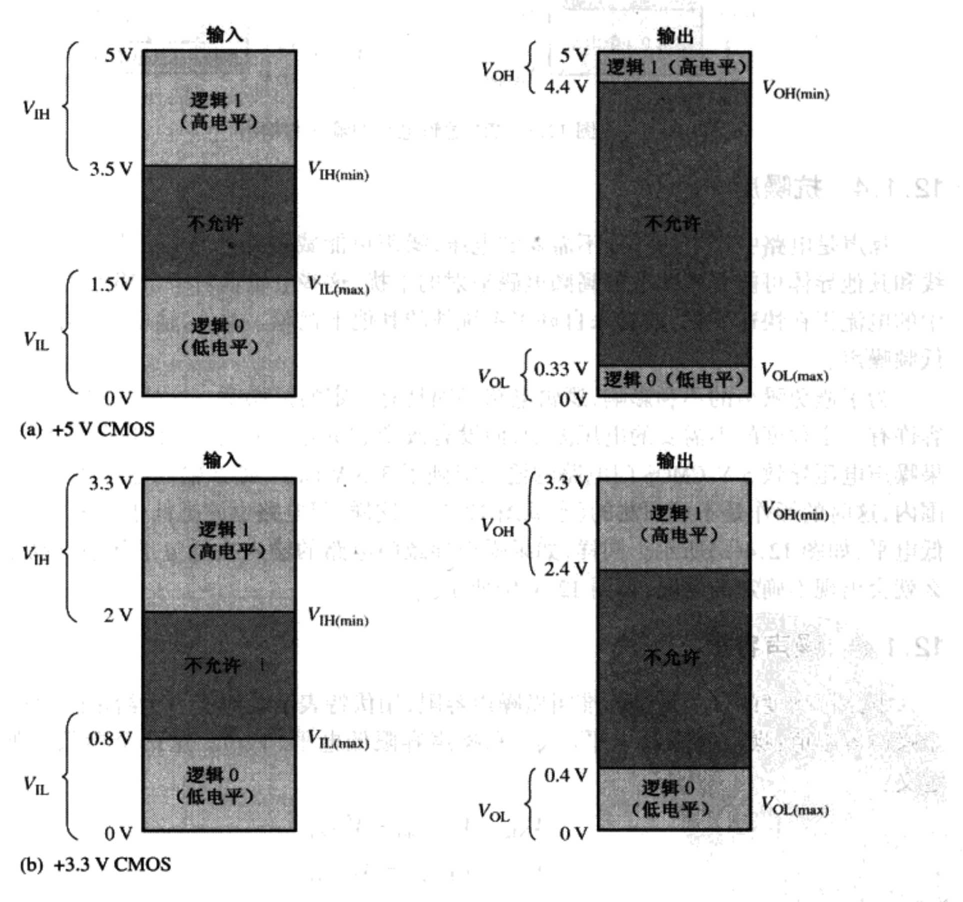


图 12.2 CMOS 的输入和输出逻辑电平

12.1.3 TTL 逻辑电平

TTL 电路的输入和输出逻辑电平如图 12.3 所示。与 CMOS 类似,它也有 4 种不同的逻辑电平规范:V_{IL},V_{IH},V_{OL},V_{OH}。

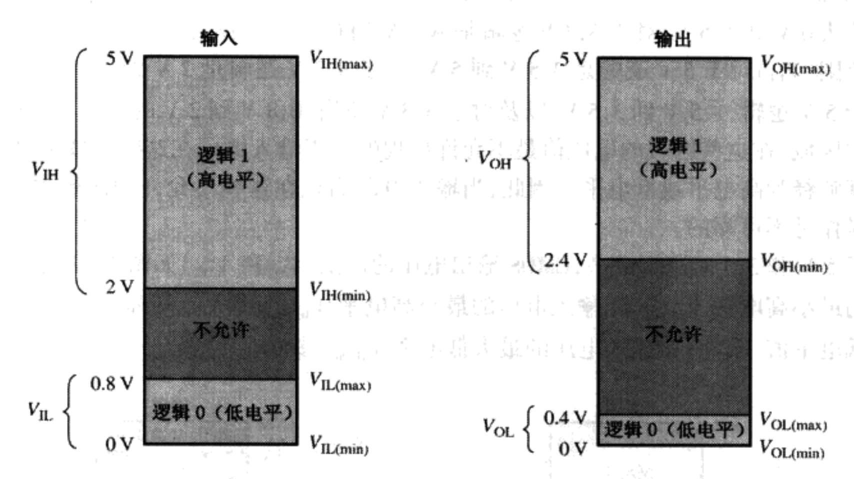


图 12.3 TTL 逻辑电平的输入和输出

12.1.4 抗噪度

噪声是电路中产生的一种不需要的电压,噪声可能威胁到电路的正常工作。系统中的导线和其他导体可能受到杂散的高频电磁辐射的干扰,这些电磁辐射来自邻近的导体,这些导体中的电流正在快速变化,或者来自许多系统外的其他干扰源。此外,输电线的电压波动是一种低频噪声。

为了避免噪声的不利影响,逻辑电路必须具有一定的抗噪度。这是一种能力,即在输入端容许有一定程度的不需要的电压波动,而没有改变它的输出状态。例如,在高电平状态下,如果噪声电压导致 5 V CMOS 门电路的输入降到了 3.5 V 以下,那么输入电压就会在不允许的范围内,这时的操作是不可预测的(参见图 12.2)。这样,门电路也许把低于 3.5 V 的波动解释为低电平,如图 12.4(a)所示。同样,如果噪声导致门电路的输入在低电平状态下高于 1.5 V,那么就会出现不确定的情况,如图 12.4(b)所示。

12.1.5 噪声容限

电路抗噪度的一个衡量标准叫做噪声容限,用伏特表示。对于一个给定的逻辑电路,有两个噪声容限值:噪声容限高电平(V_{NH})和噪声容限低电平(V_{NL})。这两个参数由如下的公式定义:

$$V_{NH} = V_{OH(min)} - V_{IH(min)} \tag{12.1}$$

$$V_{NL} = V_{IL(max)} - V_{OL(max)} \tag{12.2}$$

有时看到噪声容限是用 V_{CC}的百分比表示的。从上面公式可以看出,V_{NH}是电压差,是驱动门高电平输出的最小值($V_{OH(min)}$)和负载门能够允许的高电平输入的最小值($V_{IH(min)}$)之差。噪声

容限 V_{NL} 是门电路能够容许的低电平输入的最大值（$V_{IL(max)}$）和驱动门电路可能的低电平输出的最大值（$V_{OL(max)}$）两者之差。噪声容限如图 12.5 所示。

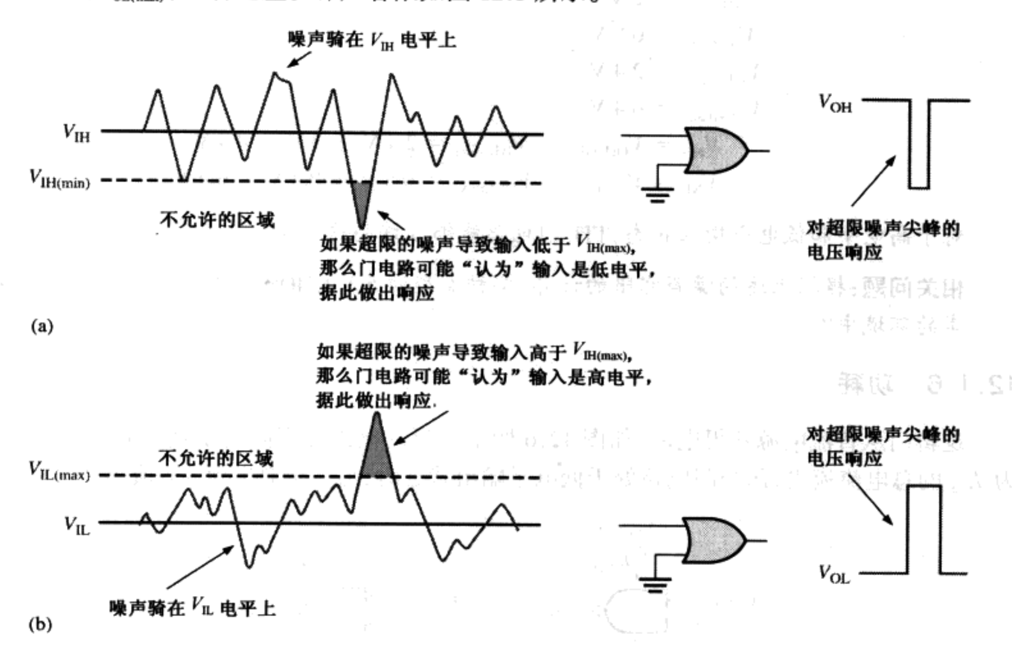


图 12.4 门电路操作中输入噪声的影响示意图

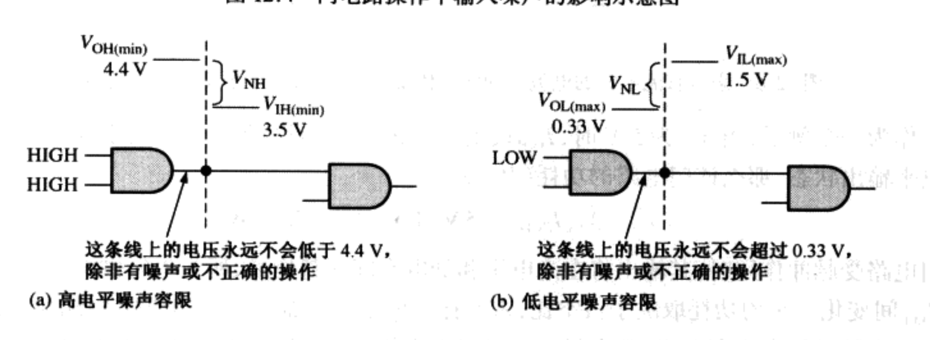


图 12.5 噪声容限的图例。以上是 5V CMOS 的值，但基本原理适用于所有系列的逻辑电路

例 12.1 使用图 12.2 和图 12.3 中的数据，确定 CMOS 电路和 TTL 电路的高电平和低电平噪声容限。

解： 对于 5 V CMOS 电路，

$$V_{IH(min)} = 3.5\text{ V}$$
$$V_{IL(max)} = 1.5\text{ V}$$
$$V_{OH(min)} = 4.4\text{ V}$$
$$V_{OL(max)} = 0.33\text{ V}$$
$$V_{NH} = V_{OH(min)} - V_{IH(min)} = 4.4\text{ V} - 3.5\text{ V} = \mathbf{0.9\text{ V}}$$
$$V_{NL} = V_{IL(max)} - V_{OL(max)} = 1.5\text{ V} - 0.33\text{ V} = \mathbf{1.17\text{ V}}$$

对于 TTL 电路,

$$V_{IH(min)} = 2\text{ V}$$
$$V_{IL(max)} = 0.8\text{ V}$$
$$V_{OH(min)} = 2.4\text{ V}$$
$$V_{OL(max)} = 0.4\text{ V}$$
$$V_{NH} = V_{OH(min)} - V_{IH(min)} = 2.4\text{ V} - 2\text{ V} = \mathbf{0.4\ V}$$
$$V_{NL} = V_{IL(max)} - V_{OL(max)} = 0.8\text{ V} - 0.4\text{ V} = \mathbf{0.4\ V}$$

对于高电平和低电平输入状态,TTL 门电路能够容许高达 0.4 V 的噪声。

相关问题:根据上述的噪声容限的计算,哪种类型的 5 V CMOS 或 TTL 芯片应该用在高噪声的环境中?

12.1.6 功耗

逻辑门从直流电源获得电流,如图 12.6 所示。当门电路处于高电平输出状态时,一个称为 I_{CCH}的总电流流出;而当门电路处于低电平输出状态时,另一个总电流 I_{CCL}流出。

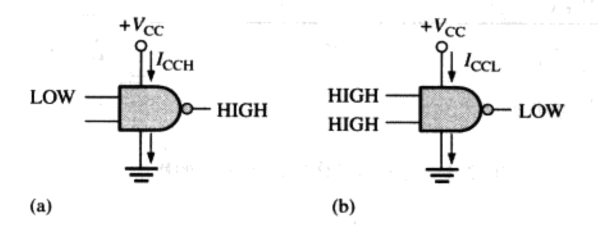


图 12.6 来自直流电源的电流。如图为传统的电流方向,电子流向的标记相反

作为一个例子,当 V_{CC}为 5 V 时,I_{CCH}设定为 1.5 mA,并且如果门电路处于静态(不变化)的高电平输出状态,那么该门电路的功耗(P_D)为

$$P_D = V_{CC}I_{CCH} = (5\text{ V})(1.5\text{ mA}) = 7.5\text{ mW}$$

当门电路受脉冲作用时,其输出将在高电平和低电平间交替切换,所产生的供电电流将在 I_{CCH}和 I_{CCL}间变化。平均功耗取决于占空比,占空比一般指定为 50%。当占空比为 50%时,那么输出在一半的时间内为高电平,而在另一半时间内为低电平。因此平均的供电电流为

$$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2} \tag{12.3}$$

平均功耗为

$$P_D = V_{CC}I_{CC} \tag{12.4}$$

例 12.2 给定一个门电路,当输出为高电平时,流入 2 μA 的电流;当输出为低电平时,流入 3.6 μA 的电流。如果 V_{CC}是 5 V,该门电路的占空比是 50%,那么平均功耗是多少?

解:平均电流 I_{CC}为

$$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2} = \frac{2.0\ \mu\text{A} + 3.6\ \mu\text{A}}{2} = 2.8\ \mu\text{A}$$

平均功耗为

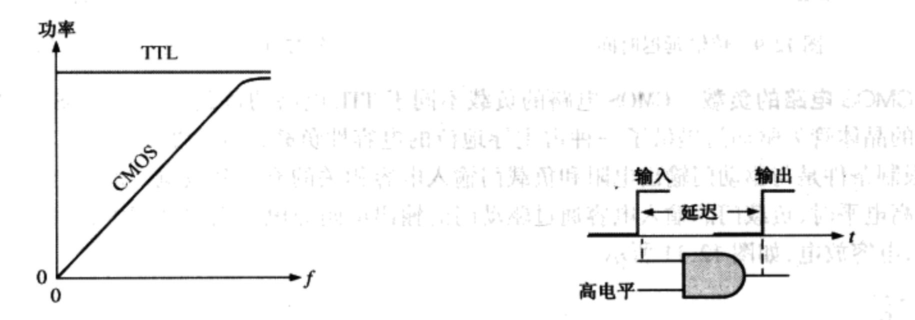
$$P_D = V_{CC}I_{CC} = (5\ V)(2.8\ \mu A) = \mathbf{14\ \mu W}$$

相关问题：一个 IC 门电路的 $I_{CCH} = 1.5\ \mu A$ 和 $I_{CCL} = 2.8\ \mu A$。确定在 V_{CC} 是 5 V、占空比是 50%的情况下，门电路的平均功耗是多少？

TTL 电路的功耗在工作频率的范围以内基本上是恒定的。但是，CMOS 电路的功耗则和频率有关。在静态(直流)的条件下它的功耗是非常小的，并随着频率的增加而增加。这些特性可对照图 12.7 中的总的曲线。例如，一个低功率肖特基(LS)TTL 门电路的功耗是常量2.2 mW。一个 HCMOS 门电路的功耗在静态条件下是 2.75 μW，在频率为 100 kHz 时是 170 μW。

12.1.7 传输延迟时间

当一个信号在逻辑电路中通过(传送)时，总会经历一个时间延迟，如图 12.8 所示。输入电平的变化引起输出电平的变化，输出电平变化的发生总是迟后输入一个很短的时间，这称为传输延迟时间。



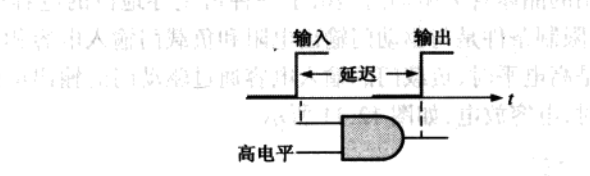


图 12.7 TTL 和 CMOS 功率 - 频率曲线

图 12.8 传输延迟时间的基本图例

正如第 3 章所提及的，逻辑门规定了两个传输延迟时间：

- t_{PHL}：当输出从高电平向低电平变化时，输入脉冲的指定点与输出脉冲的对应点之间的时间。
- t_{PLH}：当输出从低电平向高电平变化时，输入脉冲的指定点与输出脉冲的对应点之间的时间。

这两种传输延迟时间如图 12.9 所示，以脉冲边沿的 50%的点作为参照点。

门电路的传输延迟时间限制了它所能工作的频率。传输延迟时间越长，最高频率就越低。因此，速度较高的电路的传输延迟时间就较短。例如，传输延迟 3 ns 的门电路就比传输延迟 10 ns 的门电路的速度快。

12.1.8 速度 - 功耗乘积

在为某种应用选择所使用的逻辑电路类型时，如果传输延迟时间和功耗都是重要的考虑因素，那么速率 - 功耗乘积提供了各种逻辑电路的比较基础。速度 - 功耗乘积越低越好。速度 - 功耗乘积的单位是微微焦耳(pJ)。例如，HCMOS 电路在 100 kHz 的频率下，其速度 - 功耗乘积为 1.2 pJ，而对应的 LS TTL 的速度 - 功耗乘积则为 22 pJ。

12.1.9 负载和扇出

当一个逻辑门的输出连接到了一个或多个其他的门的输入时,驱动门上就会有一个负载,如图 12.10 所示。对于一个给定的门,它可以驱动的负载门输入是有数量限制的。这个限制就称为这个门的扇出。

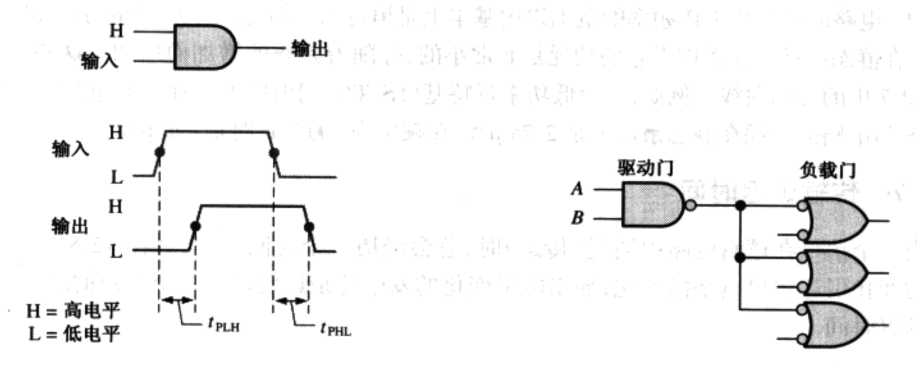


图 12.9 传输延迟时间

图 12.10 带门输入负载的门输出

CMOS 电路的负载 CMOS 电路的负载不同于 TTL 电路的负载,因为 CMOS 逻辑电路所使用的晶体管为驱动门提供了一种占主导地位的电容性负载,如图 12.11 所示。在这种情况下,限制条件是与驱动门输出电阻和负载门输入电容相关的充电和放电时间。当驱动门的输出是高电平时,负载门的输入电容通过驱动门的输出电阻充电。当驱动门电路的输出是低电平时,电容放电,如图 12.11 所示。

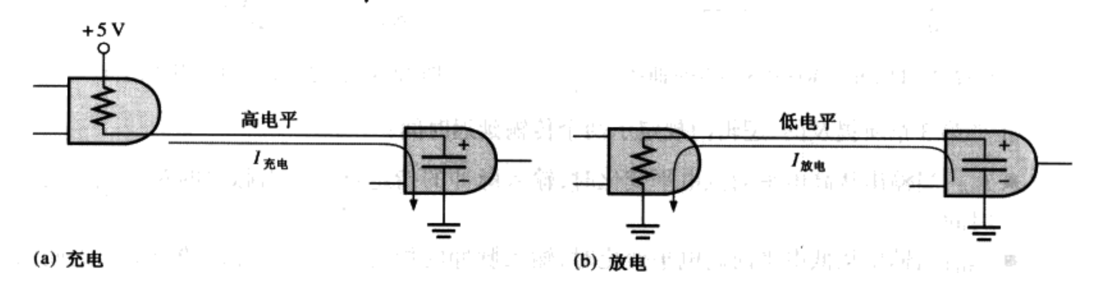


图 12.11 CMOS 门的容性负载

如果在驱动门的输出添加更多的负载门输入,那么总的电容将会增加,因为输入电容实际上是并行出现的。电容的增加会延长充电和放电时间,因此就会降低这个门电路的最大工作频率。因此,CMOS 门电路的扇出是和工作频率有关的。负载门输入越少,最大频率就越高。

TTL 负载 TTL 驱动门处于高电平状态时向负载门提供灌电流(I_{IH}),处于低电平状态时,从负载门吸收拉电流(I_{IL})。灌电流(current sourcing)和拉电流(current sinking)如图 12.12 所示,其中的电阻代表两种情况下门内部的输入和输出电阻。

当更多的负载门连接到驱动门时,驱动门电路上的负载就会增加。总的灌电流随着每增加一个门输入而增加,如图 12.13 所示。随着电流的增加,驱动门内部的电压降也会增加,从

而引起输出电压 V_{OH} 的降低。如果连接过多的负载门输入，V_{OH} 就会降到低于 $V_{OH(min)}$，高电平噪声容限也会降低，从而危及到电路的正常工作。同样，随着灌电流的增加，驱动门的功耗也会增加。

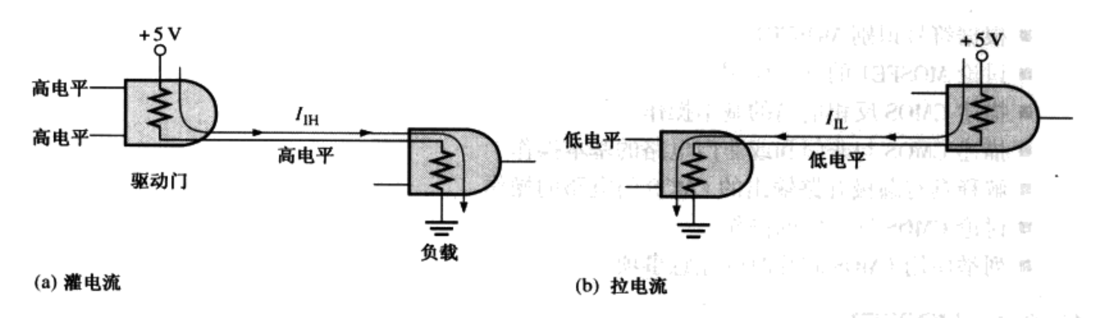


图 12.12　逻辑门中灌电流和拉电流的基本图例

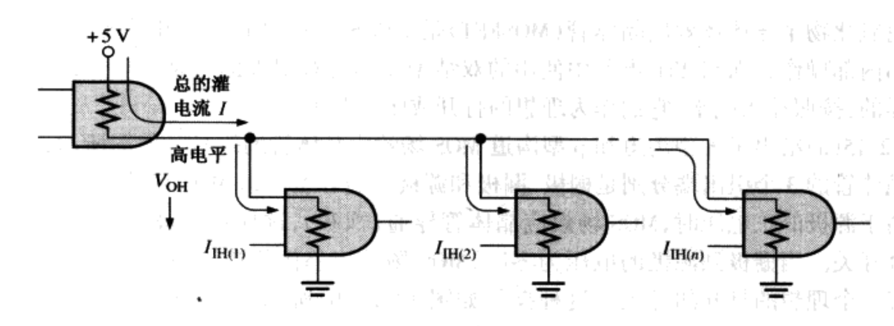


图 12.13　高电平状态下的 TTL 负载

扇出是指在不会对门的工作特性造成不利影响的情况下，门所能够连接的负载门输入的最大数目。例如，低功率肖特基(LS)TTL 电路的扇出是 20 个单位负载。作为驱动电路的相同逻辑系列的一个输入称为一个单位负载。

总的拉电流也会随着每增加一个负载门输入而增加，如图 12.14 所示。当拉电流增加时，驱动门电路内部的电压降也会增加，从而导致 V_{OL} 增加。如果添加了过多的负载，V_{OL} 将会超过 $V_{OL(max)}$，而低电平噪声容限会减小。

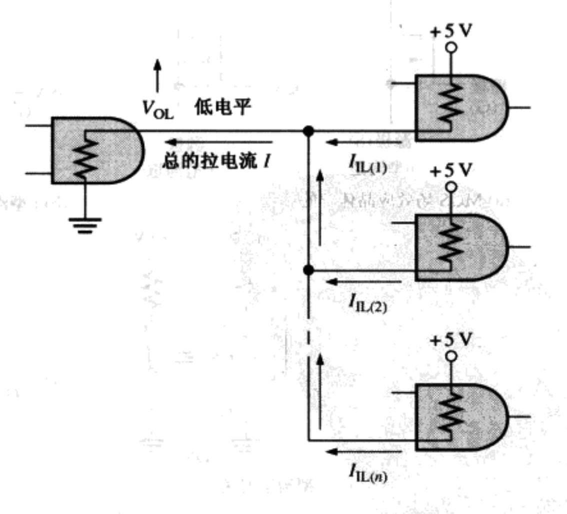


图 12.14　低电平状态时的 TTL 负载

在 TTL 电路中，拉电流的能力(低电平输出状态)是决定扇出数量的限制因素。

12.2　CMOS 电路

本节将讨论基本的 CMOS 的内部电路及其操作。缩写 CMOS 表示“互补金属氧化物半导体”。术语“互补”是指在输出电路中

使用了两种类型的晶体管,即使用了一个 n 型沟道的 MOSFET(MOS 场效应晶体管)和使用一个 p 型沟道的 MOSFET。

学完本节以后,应当能够

- 根据符号识别 MOSFET
- 讨论 MOSFET 的开关作用
- 描述 CMOS 反相电路的基本操作
- 描述 CMOS 与非门和或非门电路的基本操作
- 解释具有漏极开路输出的 CMOS 门电路的操作
- 讨论 CMOS 三态门的操作
- 列举使用 CMOS 芯片时的注意事项

12.2.1 MOSFET

金属氧化物半导体场效应晶体管(MOSFET)是 CMOS 电路中的有源开关元件。这些芯片在结构和内部操作方面与 TTL 电路中使用的双结型晶体管有很大的差别,但开关的作用基本上是一样的:按照输入电平,它们作为理想的打开或闭合开关。

图 12.15(a)给出了 n 型沟道和 p 型沟道 MOS 场效应晶体管的符号。如图所示,一个 MOS 场效应晶体管的 3 个引出端分别是栅极、漏极和源极。当 n 型沟道 MOS 场效应晶体管栅极的正电压高于源极的正电压时,MOS 场效应晶体管导通(饱和),这样在漏极和源极间有一个理想的闭合开关。当栅极到源极的电压为零时,MOS 场效应晶体管断开(截止),这样在漏极和源极间有一个理想的打开的开关。这种操作如图 12.15(b)所示。p 型沟道 MOS 场效应晶体管在相反的电压极性下工作,如图 12.15(c)所示。

有时使用简化的 MOS 场效应晶体管符号,如图 12.16 所示。

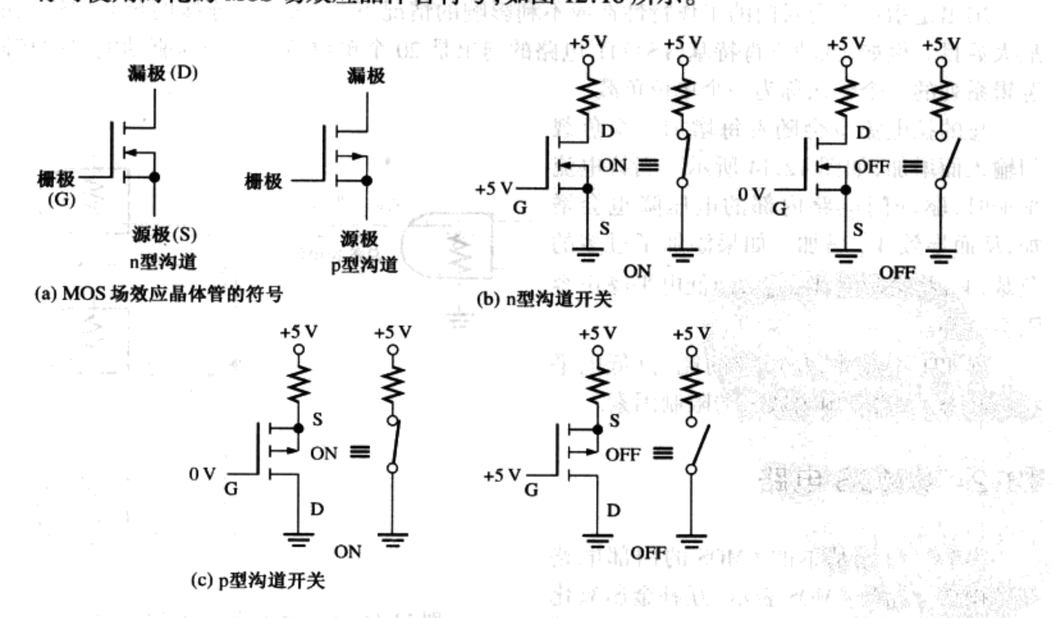


图 12.15 MOS 场效应晶体管的基本符号和开关作用

12.2.2 CMOS 反相器

互补 MOS(CMOS)逻辑把互补对中的 MOS 场效应晶体管用做基本单元。一个互补对使用 p 型沟道和 n 型沟道增强型 MOS 场效应晶体管,如图 12.17 中的反相电路。

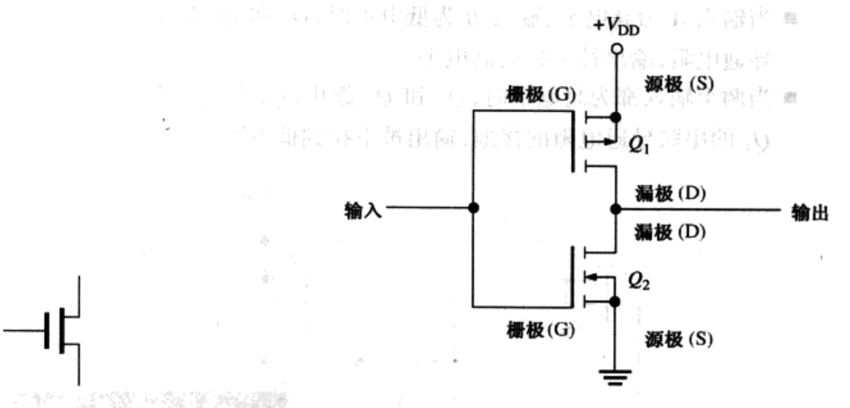


图 12.16　简化的 MOS 场效应晶体管符号

图 12.17　一个 MOS 反相器电路

当一个高电平加在输入时,如图 12.18(a)所示,p 型沟道 MOS 场效应晶体管 Q_1 截止,n 型沟道 MOS 场效应晶体管 Q_2 导通。这个情况通过 Q_2 的导通电阻使输出连接到地,导致低电平输出。当一个低电平加在输入时,如图 12.18(b)所示,Q_1 导通,Q_2 截止。这个情况通过 Q_1 的导通电阻把输出连接到 + V_{DD},导致高电平输出。

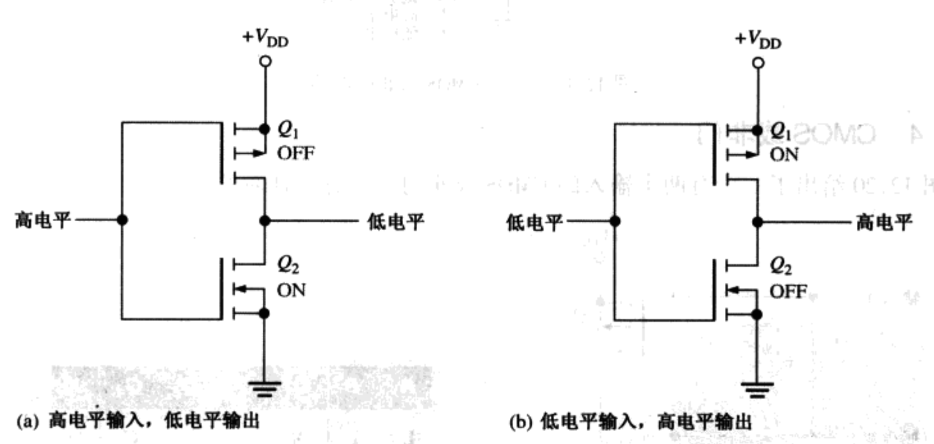


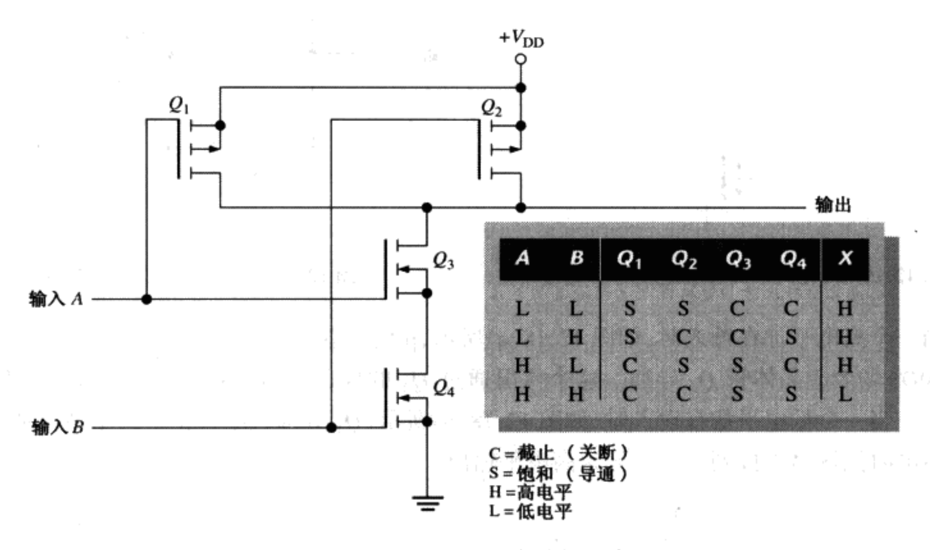
图 12.18　CMOS 反相器的工作原理

12.2.3 CMOS 与非门电路

图 12.19 给出了一个有两个输入的 CMOS 与非门电路。注意互补对(n 型沟道和 p 型沟道 MOS 场效应晶体管)的排列。

CMOS 与非门电路的工作如下所述。

- 当两个输入都是低电平时，Q_1 和 Q_2 导通，Q_3 和 Q_4 截止。通过 Q_1 和 Q_2 并联的导通电阻，输出被上拉为高电平。
- 当输入 A 为低电平、输入 B 为高电平时，Q_1 和 Q_4 导通，Q_2 和 Q_3 截止。通过 Q_1 的低导通电阻，输出被上拉为高电平。
- 当输入 A 为高电平、输入 B 为低电平时，Q_1 和 Q_4 截止，Q_2 和 Q_3 导通。通过 Q_2 的低导通电阻，输出被上拉为高电平。
- 当两个输入都为高电平时，Q_1 和 Q_2 截止，Q_3 和 Q_4 导通。在这个情况下，通过 Q_3 和 Q_4 的串联导通电阻的接地，输出被下拉到低电平。

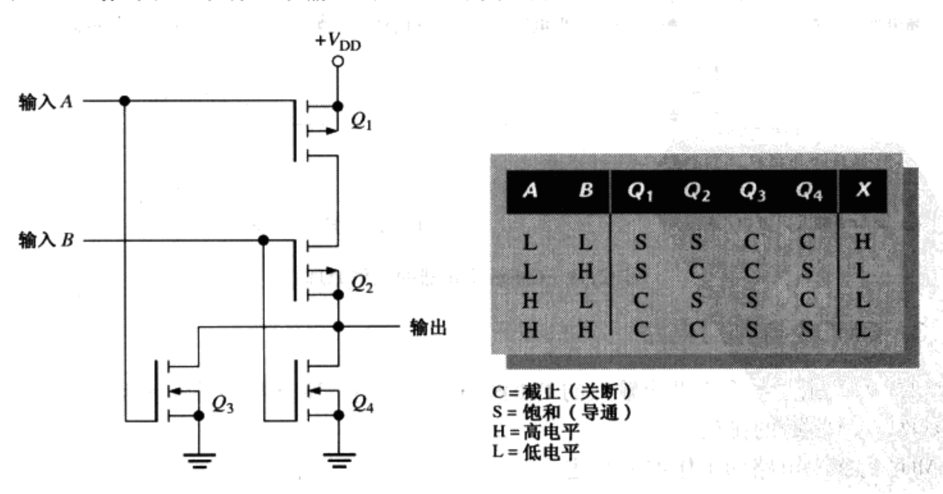


A	B	Q_1	Q_2	Q_3	Q_4	X
L	L	S	S	C	C	H
L	H	S	C	C	S	H
H	L	C	S	S	C	H
H	H	C	C	S	S	L

图 12.19 一个 CMOS 与非门电路

12.2.4 CMOS 或非门

图 12.20 给出了一个有两个输入的 CMOS 或非门。注意互补对的排列。



A	B	Q_1	Q_2	Q_3	Q_4	X
L	L	S	S	C	C	H
L	H	S	C	C	S	L
H	L	C	S	S	C	L
H	H	C	C	S	S	L

图 12.20 一个 CMOS 或非门电路

CMOS 或非门电路的操作如下所述。

- 当两个输入都是低电平时，Q_1 和 Q_2 导通，Q_3 和 Q_4 截止。结果，通过 Q_1 和 Q_2 导通电阻的串联，输出被上拉为高电平。
- 当输入 A 为低电平、输入 B 为高电平时，Q_1 和 Q_4 导通，Q_2 和 Q_3 截止。通过 Q_4 的低导通电阻接地，输出被下拉到低电平。
- 当输入 A 为高电平、输入 B 为低电平时，Q_1 和 Q_4 截止，Q_2 和 Q_3 导通。通过 Q_3 的导通电阻接地，输出被下拉到低电平。
- 当两个输入都是低电平时，Q_1 和 Q_2 截止，Q_3 和 Q_4 导通。通过 Q_3 和 Q_4 并联的导通电阻接地，输出被下拉到低电平。

12.2.5　漏极开路门

术语“漏极开路”的意思是输出晶体管的漏极端没有被连接，而必须从外部通过负载连接到 V_{DD}。漏极开路门是与集电极开路 TTL 门(下一节讨论)相类似的 CMOS 芯片。漏极开路输出电路是一个单一的 n 型沟道 MOS 场效应晶体管，如图 12.21(a)所示。为了产生高电平输出，必须使用一个外部上拉电阻，如图 12.21(b)所示。此外，漏极开路输出能够以线与的方法连接，这个概念在下一节涉及 TTL 时讨论。

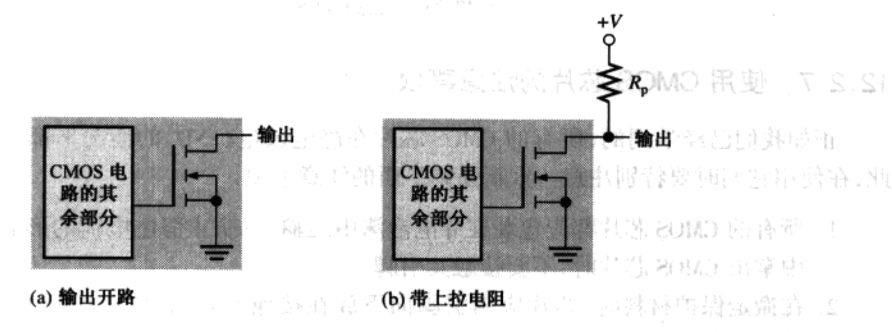


图 12.21　漏极开路 CMOS 门

12.2.6　CMOS 三态门

三态输出在 CMOS 和 TTL 逻辑电路中都会出现。三态输出结合了推拉输出和集电极开路电路的优点。回顾一下，三种输出状态是高电平、低电平和高阻抗(高 Z)。由使能($\overline{\text{Enable}}$)输入的状态决定三态门的输出状态，当选择工作在正常的逻辑电平时，三态电路和常规门的工作方式相同。当三态电路选择工作在高阻状态时，通过内部电路把三态电路的输出和其余部分有效地断开。图 12.22 给出了三态电路的操作。下三角(▽)表示三态输出。

在三态 CMOS 门的电路系统中，如图 12.23 所示，允许把每个输出晶体管 Q_1 和 Q_2 同时关闭，从而断开了输出和电路其余部分的连接。

当使能输入为低电平时，使得电路工作在正常的逻辑状态。当使能端入为高电平时，Q_1 和 Q_2 都截止，电路处于高阻状态。

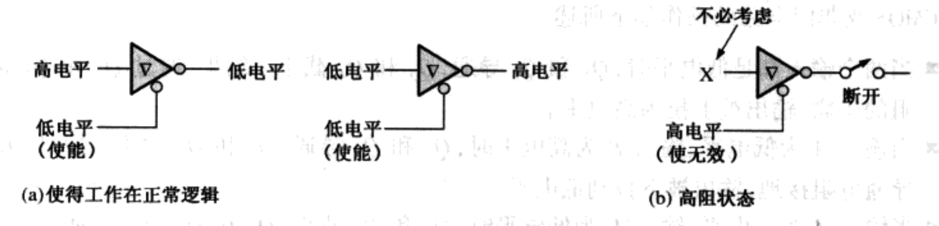


图 12.22 三态电路的三个状态

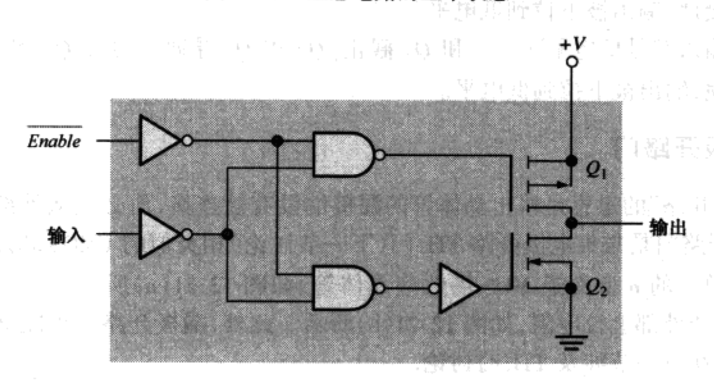


图 12.23 三态 CMOS 反相器

12.2.7 使用 CMOS 芯片的注意事项

正如我们已经学到的,所有的 CMOS 芯片在静电放电(ESD)的情况下都会遭受损害。因此,在使用它们时要特别注意。重新强调下面的注意事项:

1. 所有的 CMOS 芯片都应包装在导电泡沫中运输,以防止静电电荷的形成。从泡沫包装中拿出 CMOS 芯片时,不要接触其引脚。
2. 在撤走保护材料时,芯片应当引脚向下放在接地的表面,如金属板上。不要将 CMOS 设备放在聚苯乙烯泡沫或塑料盘上。
3. 所有的工具、测试设备和金属工作台都应当接地。在某些环境中,使用 CMOS 芯片的人员应当在手腕上带有一段电缆并串联一个高值电阻接地。这个电阻将在操作人员接触电压源时防止强烈的电击。
4. 不要把 CMOS 芯片(或任何其他 IC)插入通电的插槽或 PC 板。
5. 所有不使用的输入都应该连接到电源或地,如图 12.24所示。如果不接,输入就可能获得静电电荷,从而在不可预测的电平上"漂浮"。
6. 在 PC 板上装配好后,在存储或运输时,把板的连接口插在泡沫中,以提供必要的保护。用高阻值的电阻把 CMOS 的引脚与地连接,这样引脚也可以得到保护。

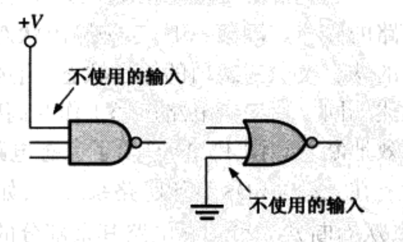


图 12.24 对 CMOS 不使用的输入的处理

12.3 TTL 电路

本节介绍带有推拉输出的 TTL 逻辑门内部电路的工作原理。此外，还将介绍集电极开路输出的 TTL 门及三态门的工作原理。

学完本节以后，应当能够

- 根据符号识别双结型晶体管(BJT)
- 描述双结型晶体管的开关特性
- 描述 TTL 反相器电路的基本工作原理
- 描述 TTL、与门、与非门、或门和或非门电路的基本原理
- 解释什么是推拉式输出
- 解释集电极开路输出的 TTL 门的工作原理和使用方法
- 解释三态输出门的工作原理

12.3.1 双结型晶体管(BJT)

双结型晶体管(BJT)是所有 TTL 电路中使用的有源开关元件。图 12.25 所示为一个 npn 双结型晶体管的符号，它有 3 个引出端：基极、发射极和集电极。双结型晶体管有两个结：基极 - 发射极结和基极 - 集电极结。

基本的开关工作原理如下：当基极的正电压高于发射极电压 0.7 V 左右，有足够的电流提供给基极，这时晶体管导通，进入饱和状态。在饱和状态下，在集电极和发射极之间，晶体管的作用就像一个理想的闭合的开关，如图 12.26(a)所示。当基极的正电压大于发射极电压的值小于 0.7 V 时，晶体管截止，在集电极和发射极之间变成一个打开的开关，如图 12.26(b)所示。用通常的术语总结，就是基极上的高电平使晶体管导通，使得它成为一个闭合开关；基极上的低电平使晶体管截止，使得它成为一个打开的开关。在 TTL 电路中，一些双结型晶体管有多个发射极。

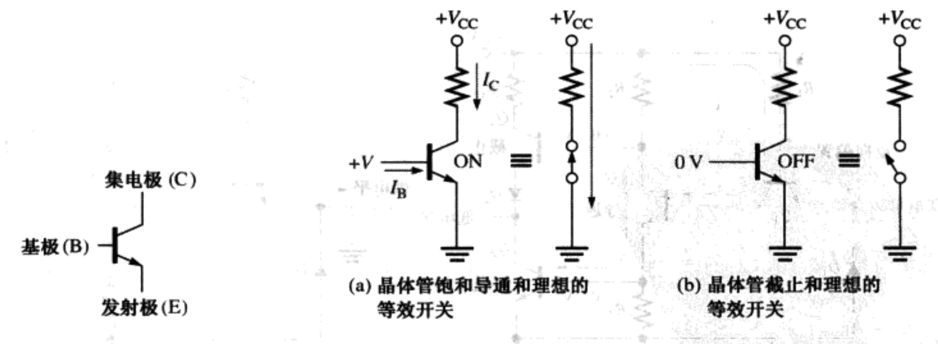


图 12.25 双结型晶体管的符号

图 12.26 晶体管理想的开关特性。传统的电流方向如图所示。电子流动方向的标记相反

12.3.2 TTL 反相器

无论采用哪种电路技术、反相器或者任何类型的门，其逻辑功能总是相同的。图 12.27 给

出了反相器的标准 TTL 电路。在图中 Q_1 是输入耦合晶体管,D_1 是输入钳位二极管。晶体管 Q_2 称为"分相器",Q_3 和 Q_4 的组合形成输出电路,通常称为推拉输出(图腾柱, totem-pole)。

当输入为高电平时,Q_1 的基极-发射极结是反向偏置,而基极-集电极结点是正向偏置。这个情况允许电流通过 R_1 和 Q_1 的基极-集电极结进入 Q_2 的基极,从而驱动 Q_2 进入饱和状态。结果,Q_2 的导通使得 Q_3 导通,Q_3 的集电极电压就是输出,它接近于接地电压。这样一个高电平输入得到一个低电平输出。同时,Q_2 的集电极处在足够低的电压以使 Q_4 保持截止。

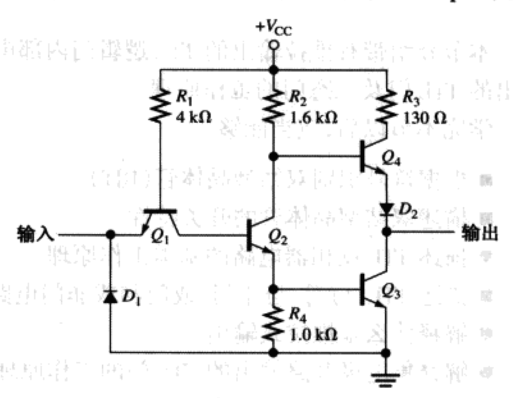


图 12.27 标准 TTL 反相器电路

当输入为低电平时,Q_1 的基极-发射极结是正向偏置,而基极-集电极结是反向偏置。电流通过 R_1 和 Q_1 的基极-发射极结进入低电平输入。低电平为电流接地提供了通路。Q_2 的基极没有电流进入,因此 Q_2 截止。Q_2 的集电极是高电平,从而使 Q_4 导通。饱和导通的 Q_4 提供了从 V_{CC}到输出的低电阻通路;因此得到一个低电平输入对应一个高电平输出。同时,Q_2 的发射极处在接地电位,保持 Q_3 截止。

TTL 电路中的二极管 D_1 防止输入负的尖峰电压,以避免损坏 Q_1。二极管 D_2 确保 Q_2 导通(高电平输入)时,Q_4 截止。在这种情况下,Q_2 的集电极电压等于 Q_3 的基极-发射极电压 V_{BE}加上 Q_2 的集电极-发射极电压 V_{CE}。二极管 D_2 提供了附加的等效电压降 V_{BE},这个等效电压和 Q_4 的基极-发射极结电压串联以保证 Q_2 导通时,Q_4 截止。

TTL 反相器在这两个输入状态下的工作如图 12.28 所示。在图 12.28(a)的电路中,Q_1 的基极对地电压是 2.1 V,因此 Q_2 和 Q_3 导通。在图 12.28(b)的电路中,Q_1 的基极对地电压是 0.7 V,不足以使 Q_2 和 Q_3 导通。

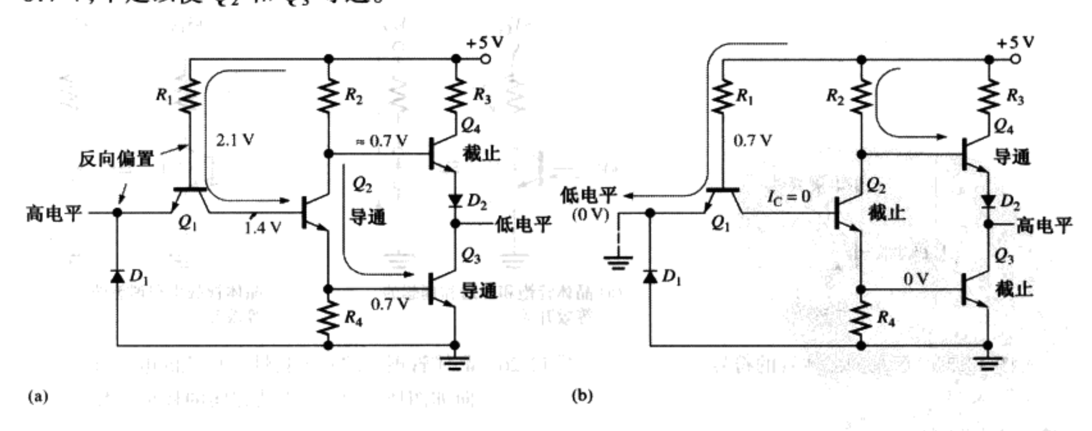


图 12.28 TTL 反相器的工作原理

12.3.3 TTL与非门

如图12.29所示是一个二输入TTL与非门。它基本上与反向器电路相同,除了增加了Q_1的发射极输入。在TTL技术中,输入芯片使用了多发射极晶体管。这些多发射极晶体管可比做二极管的排列,如图12.30所示。

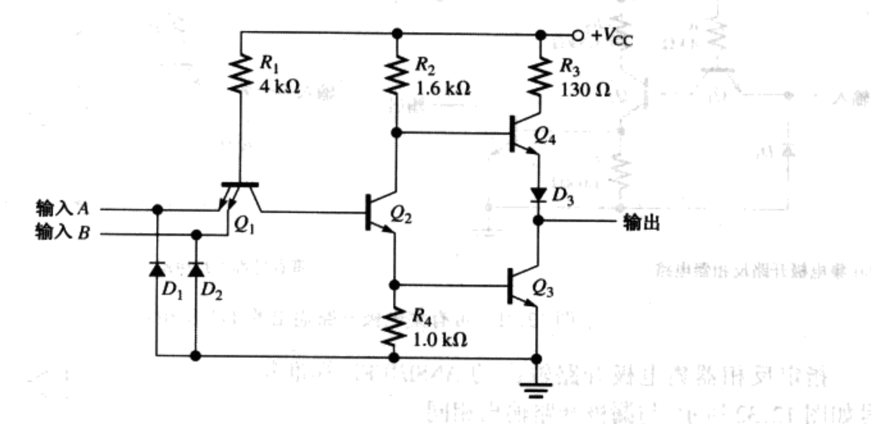


图12.29 TTL与非门电路

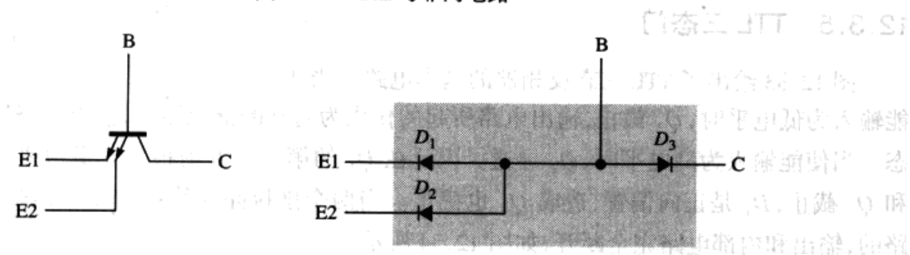


图12.30 TTL多发射极晶体管的二极管等效电路

把图12.29中的Q_1具体化,用图12.30中的二极管的连接替代Q_1,也许这样就可以更好地理解这个电路的工作原理。输入A或输入B上的低电平使各自的二极管正向偏置,使D_3(Q_1的基极-集电极结)反向偏置。这个反向偏置使Q_2保持截止,和所描述的TTL反相器相同,最后导致高电平输出。当然,两个输入同时为低电平也会产生相同的结果。

两个输入都是高电平使得两个输入二极管都处在反向偏置,D_3(Q_1的基极-集电极结)正向偏置。这个正向偏置使Q_2导通,和所描述的TTL反相器相同,最后导致低电平输出。可以把以上的分析结果看做与非门的函数关系:当且仅当所有的输入都是高电平时,输出为低电平。

12.3.4 集电极开路门

前面章节介绍的TTL门电路都具有推拉(totem-pole)输出电路。TTL集成电路中可以使用的另一种类型的输出电路是集电极开路输出。它可以与CMOS的漏极开路输出相比。具有集电极开路的标准的TTL反相器如图12.31(a)所示。还有一些其他类型的门电路也具有集电极开路输出。

注意，晶体管 Q_3 的集电极是输出，没有连接任何东西，因此取名为"集电极开路"。为了在电路的输出得到正确的逻辑高电平和低电平，必须在 Q_3 的集电极到 V_{CC} 之间连接一个外部上拉电阻，如图 12.31(b)所示。当 Q_3 截止时，输出通过外部电阻被上拉到 V_{CC}。当 Q_3 导通时，输出通过饱和导通的晶体管连接到附近的地线。

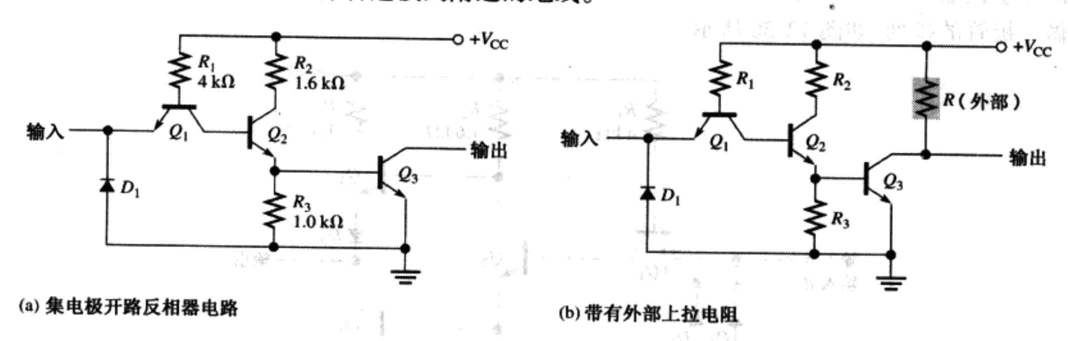


图 12.31　带有集电极开路输出的 TTL 反相器

指定反相器集电极开路输出的 ANSI/IEEE 标准符号如图 12.32 所示，与漏极开路输出相同。

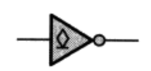
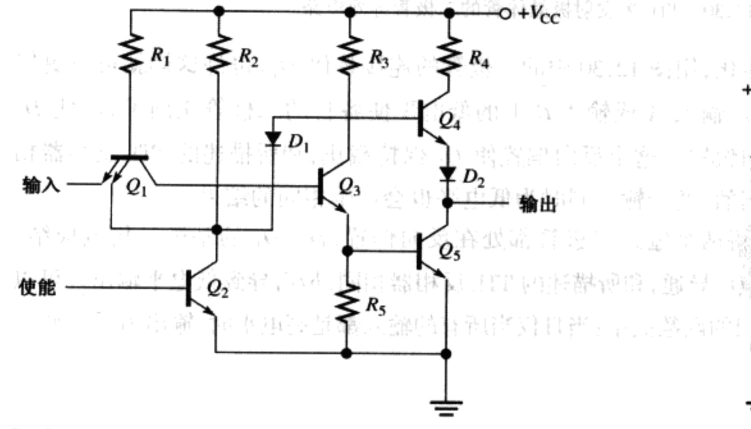


图 12.32　一个反相器中的集电极开路符号

12.3.5　TTL 三态门

图 12.33 给出了 TTL 三态反相器的基本电路。当使能输入为低电平时，Q_2 截止，输出电路所起的作用为通常的推拉输出，输出状态依赖于输入状态。当使能输入为高电平时，Q_2 导通。因此在 Q_1 的第二个发射极上是低电平，从而使得 Q_3 和 Q_5 截止，D_1 是正向偏置，造成 Q_4 也截止。当两个推拉晶体管都截止时，它们实际上是开路的，输出和内部电路完全断开，如图 12.34 所示。



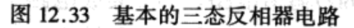


图 12.33　基本的三态反相器电路　　图 12.34　处于高阻状态的三态输出等价电路

12.3.6　肖特基 TTL

前面已经介绍了基本的或标准的 TTL 与非门电路。它是一种拉电流型的逻辑电路，当输

出状态为低电平时,电路从负载上获取电流;当输出状态为高电平时,提供给负载的电流可以忽略。现在大多数 TTL 逻辑电路都使用某些形式的肖特基 TTL,通过采纳肖特基二极管以防止晶体管进入饱和状态,使得肖特基(Schottky)TTL 具有更快的开关速度,因为减少了晶体管的开关时间。图 12.35 给出了一个肖特基门电路。请注意肖特基晶体管和肖特基二极管的符号。肖特基芯片以芯片编号中的"S"表示,如 74S00。其他一些类型的肖特基 TTL 电路,如低功耗的肖特基以 LS 表示,高级肖特基以 AS 表示,高级低功耗肖特基以 ALS 表示,快速肖特基以 F 表示。

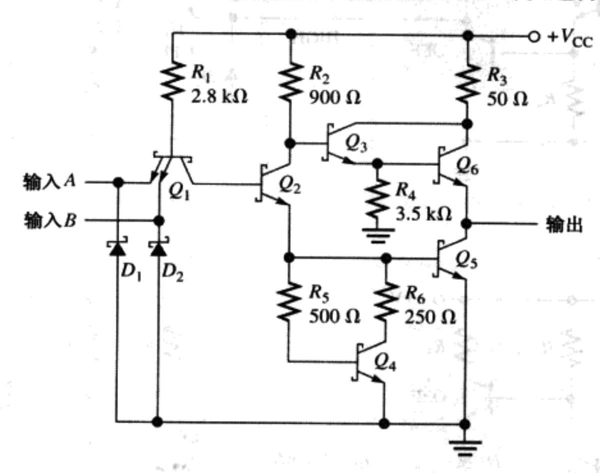


图 12.35 肖特基 TTL 与非门电路

12.4 TTL 在实际使用中的注意事项

尽管 CMOS 在工业和商业应用领域中是一种占主导地位 IC 技术,但 TTL 仍然在使用。在教育应用方面,由于静电放电(ESD)的缘故,在使用 CMOS 时有所限制,所以通常选用 TTL。正因为如此,这里以标准 TTL 为例,介绍在 TTL 电路应用中的几个注意事项。

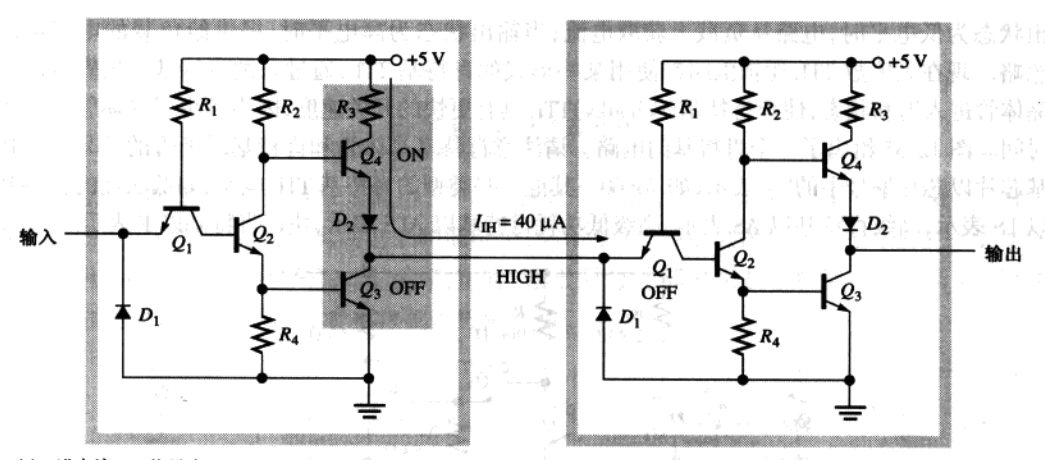
学完本节以后,应当能够

- 描述拉电流和灌电流
- 使用集电极开路电路实现线与
- 描述连接两个或多个推拉(totem-pole)输出的效果
- 用集电极开路门驱动 LED 和电灯
- 解释如何处理未使用的 TTL 输入

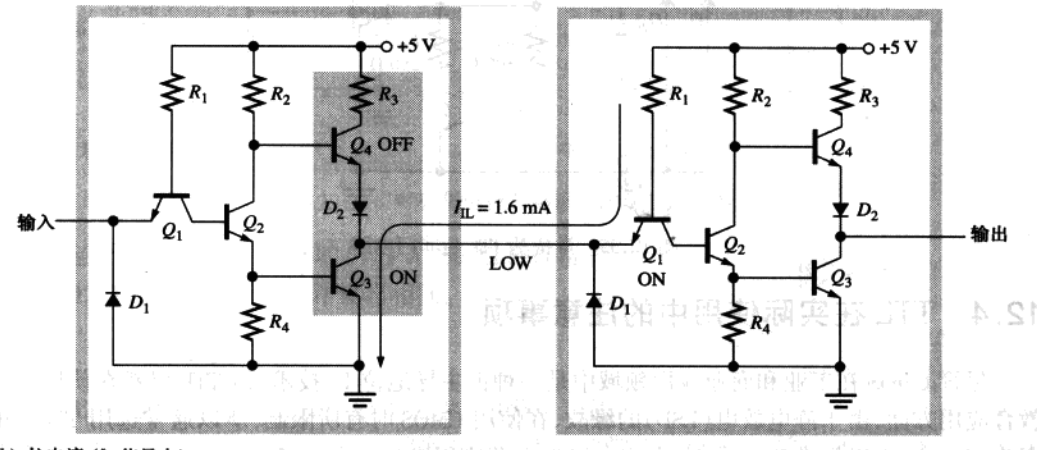
12.4.1 拉电流和灌电流

拉电流和灌电流的概念在 12.1 节中已经介绍过。现在,我们已经熟悉了在 TTL 中使用的推拉输出电路这样的结构,下面进一步了解拉电流和灌电流的形成过程。

图 12.36 给出了一个标准的 TTL 反相器,它的推拉输出连接到了另一个 TTL 反相器的输入。当驱动门电路处在高电平输出状态时,驱动器将向负载提供电流,如图 12.36(a)所示。对负载门电路的输入就像一个反向偏置的二极管,所以负载几乎不需要电流。事实上,因为输入是非理想的,所以从驱动器的推拉输出到负载门输入有 40 μA 的最大电流值。



(a) 灌电流(I_{IH}值最大)



(b) 拉电流 (I_{IL}值最大)

图 12.36 TTL 中的灌电流与拉电流

当驱动门电路处于低电平输出状态时,驱动器从负载获取电流,如图 12.36(b)所示。对于标准 TTL 电路来说,这个电流的最大值为 1.6 mA,由于超出了输入范围,所以在数据表中以负值表示。

例 12.3 当 TTL 与非门驱动 5 个 TTL 输入时,灌电流和拉电流各为多少?(参见图 12.36。)

解: 总的灌电流(高电平输出状态):

$$I_{IH(max)} = 40\ \mu A/输入$$

$$I_{T(灌电流)} = (5\ 输入)(40\ \mu A/输入) = 5(40\ \mu A) = \mathbf{200\ \mu A}$$

总的拉电流(低电平输出状态):

$$I_{IL(max)} = -1.6\ mA/输入$$

$$I_{T(拉电流)} = (5\ 输入)(-1.6\ mA/输入) = 5(-1.6\ mA) = \mathbf{-8.0\ mA}$$

相关问题: 计算 LS TTL 与非门相应的值。

前面已经介绍了基本的或标准的 TTL 与非门电路,它是一种拉电流型的逻辑电路。当输

例 12.4　参照德州仪器公司的相关数据表,确定 7400 与非门的扇出数。

解: 按照数据表,电流的参数如下:

$$I_{IH(max)} = 40\ \mu A \qquad I_{OH(max)} = -400\ \mu A$$
$$I_{IL(max)} = -1.6\ mA \qquad I_{OL(max)} = 16\ mA$$

对于高电平输出状态的扇出计算如下:电流 $I_{OH(max)}$ 是与非门可以提供给负载的最大电流。每个负载输入需要 40 μA 的电流 $I_{IH(max)}$。高电平状态的扇出为

$$\left|\frac{I_{OH(max)}}{I_{IH(max)}}\right| = \frac{400\ \mu A}{40\ \mu A} = \mathbf{10}$$

对于低电平输出状态,扇出的计算如下:电流 $I_{OL(max)}$ 是与非门可以获取的最大电流。每个负载输入产生一个 -1.6 mA 的电流 $I_{IL(max)}$。低电平状态的扇出为

$$\left|\frac{I_{OL(max)}}{I_{IL(max)}}\right| = \frac{16\ mA}{1.6\ mA} = \mathbf{10}$$

在这种情况下,高电平扇出和低电平扇出相同。

相关问题: 确定 74LS00 与非门的扇出。

12.4.2　使用集电极开路门实现线与的功能

集电极开路门的输出可连接在一起,形成一种称为“线与”(wired AND)的电路。图 12.37 给出 4 个反相器连接在一起,产生一个四输入与非门电路。在所有的线与电路中,需要一个外部上拉电阻 R_p。

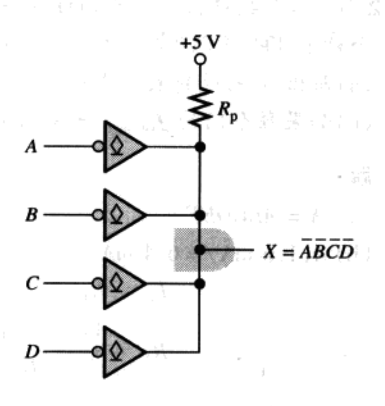


图 12.37　4 个反相器的线与连接

当一个(或多个)反相器输入为高电平时,输出 X 被拉为低电平,因为有一个输出晶体管导通,作为闭合的开关连接地,如图 12.38(a)所示。在这种情况下,只有一个反相器为高电平输入,但是如图所示,通过饱和导通的输出晶体管 Q_1 已经足够把输出拉为低电平。

对于 X 为高电平的情况,反相器所有的输入都必须为低电平,这样所有的集电极开路输出都截止,如图 12.38(b)所示。当这种情况发生时,输出 X 通过上拉电阻被拉到高电平。因此,当且仅当所有的输入为低电平时,输出 X 为高电平。因此,得到了一个与非函数,由下面公式表示:

$$X = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

例 12.5　写出图 12.39 所示的集电极开路与门的线与电路的输出表达式。

解: 输出表达式为

$$X = ABCDEFGH$$

四个 2 输入与门电路的线与创建了一个 8 输入的与门电路。

相关问题: 如果在图 12.39 中使用与非门电路,确定出输出表达式。

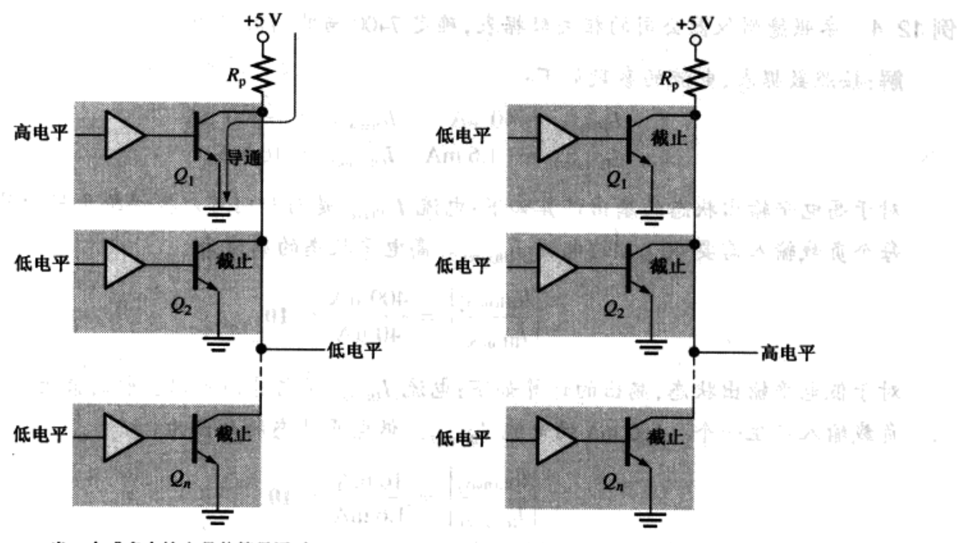


图 12.38　反相器集电极开路门的线与的工作原理

例 12.6　三个集电极开路与门连接成线与的电路，如图 12.40 所示。假定此线与电路驱动四个标准 TTL 输入(每个为 −1.6 mA)。

(a)写出 X 的逻辑表达式；

(b)如果每个门的 $I_{OL(max)}$ 是 30 mA，$V_{OL(max)}$ 是 0.4 V，确定 R_p 的最小值。

解：

(a) $X = ABCDEF$

(b) $4(1.6\text{ mA}) = 6.4\text{ mA}$

$$I_{R_P} = I_{OL(max)} - 6.4\text{ mA} = 30\text{ mA} - 6.4\text{ mA} = 23.6\text{ mA}$$

$$R_P = \frac{V_{CC} - V_{OL(max)}}{I_{R_P}} = \frac{5\text{ V} - 0.4\text{ V}}{23.6\text{ mA}} = \mathbf{195\ \Omega}$$

相关问题：给出使用 74LS09 四 − 2 输入与门组成 10 输入与函数的线与电路。

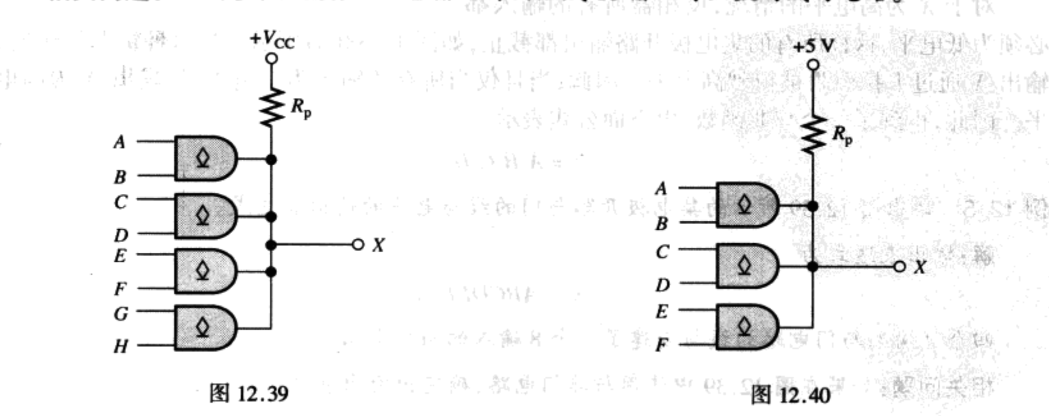


图 12.39　　图 12.40

12.4.3 推拉输出的连接

推拉输出不能连接在一起,因为这种连接可能产生过电流从而导致电路的损害。例如,在图 12.41 中,当电路 A 中的 Q_1 和电路 B 中的 Q_2 都导通时,电路 A 的输出实际上是通过电路 B 的 Q_2 短接到地的。

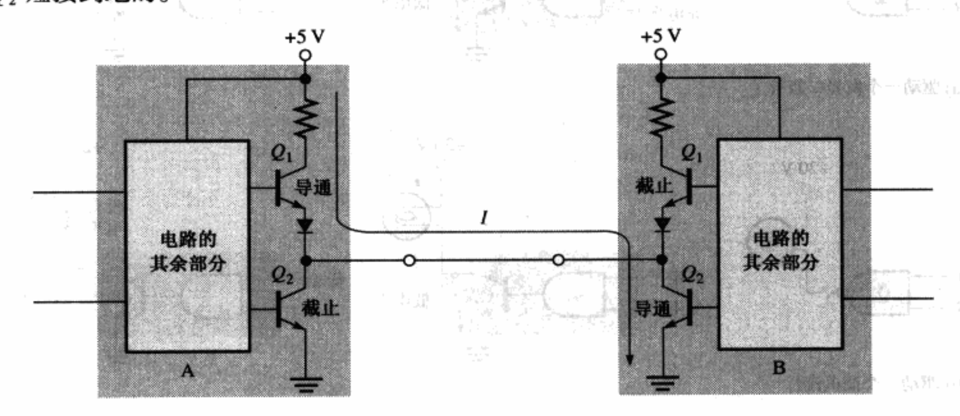


图 12.41 推拉输出线与在一起。这样的连接可能导致过电流经过电路A中的 Q_1 和电路B中的 Q_2,永远不应该这样使用

12.4.4 集电极开路缓冲器/驱动器

带有推拉输出的 TTL 电路的电流是有限制的,对于标准的 TTL 电路,在低电平状态下,它的拉电流($I_{OL(max)}$)为 16 mA,对于 LS TTL 电路则为 8 mA。在许多特殊的应用中,一个门电路必须驱动外部电路,如发光二极管(LED)、电灯或者继电器,这就需要更多的电流。

由于其更高的电压和电流的驱动能力,集电极开路输出电路通常用于驱动发光二极管、电灯或者继电器。但是,只要外部电路所需要的输出电流没有超出 TTL 驱动器能够提供的拉电流,就可以使用推拉输出。

使用集电极开路 TTL 门,输出晶体管的集电极连接到了一个发光二极管或白炽灯,如图 12.42所示。在图 12.42(a)中的限流电阻 R_L 用于确保发光二极管的电流低于最大值。当门电路的输出为低电平时,输出晶体管拉电流,发光二极管导通(点亮)。当输出晶体管截止时,输出是高电平,发光二极管不导通(熄灭)。典型的集电极开路缓冲门可以提供的拉电流高达 40 mA。在图 12.42(b)中,灯并不需要限流电阻,因为灯丝就是电阻。一般情况下,在集电极开路处可以使用高达 +30 V 的电压,可以根据特定的逻辑电路系列确定。

例 12.7 确定图 12.43 的集电极开路电路的限流电阻 R_L 的阻值,假设发光二极管的电流为 20 mA,正向偏置的电压降为 1.5 V。门电路输出电压在低电平状态时为 0.1 V。

解:

$$V_{R_L} = 5\text{ V} - 1.5\text{ V} - 0.1\text{ V} = 3.4\text{ V}$$

$$R_L = \frac{V_{R_L}}{I} = \frac{3.4\text{ V}}{20\text{ mA}} = \mathbf{170\ \Omega}$$

相关问题: 如果发光二极管需要 35 mA 的电流,确定限流电阻 R_L 的阻值。

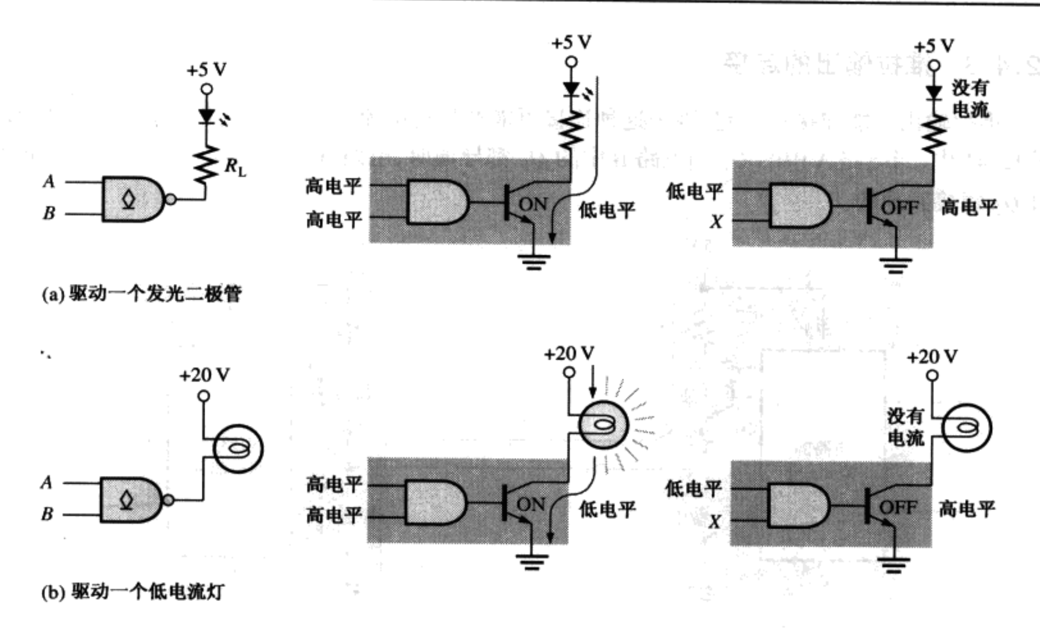


图 12.42　集电极开路驱动器的一些应用

12.4.5　未用的 TTL 输入

TTL 门中悬空的输入是作为高电平的，因为悬空的输入导致输入晶体管的发射极结反向偏置，可以达到高电平的效果。这种效应如图 12.44 所示。但是，由于对噪声的敏感，所以最好不要使未用的 TTL 输入悬空（开路）。这里有几种可选择的方法用于处理未用的输入。

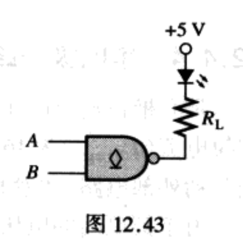


图 12.43

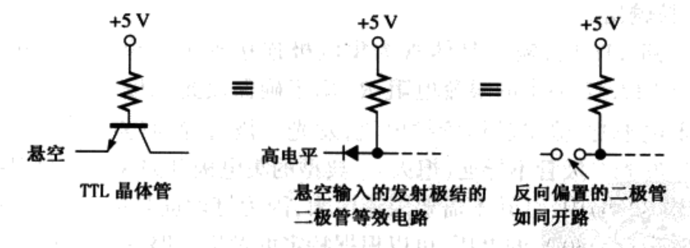


图 12.44　开路的 TTL 输入和高电平输入的比较

输入连接在一起　处理未用门电路输入的最常用的方法就是将它们连接到同一个门的一个使用的输入上。对于与门和与非门，在低电平状态下，所有连接在一起的输入看做一个单位负载；但对于或门和或非门来说，在低电平状态下，每个连到其他输入的输入都看做为各自的单位负载。在高电平状态下，对于所有类型的 TTL 门电路，连接在一起的每个输入都做作为各自的负载。在图 12.45(a)中，两个未使用的输入连接到了一个已使用的输入上。

无论多少个输入连接在一起，与门和与非门都只提供一个单位负载，而对于或门和或非门，连接在一起的每个输入都是一个单位负载。这是因为与非门使用一个多发射极输入的晶

体管;所以无论多少个输入是低电平,低电平状态的总电流都被限制为一个固定的数值。或非门电路对每个输入都使用各自的晶体管;因此,低电平状态的电流是所有连接在一起的输入电流的和。

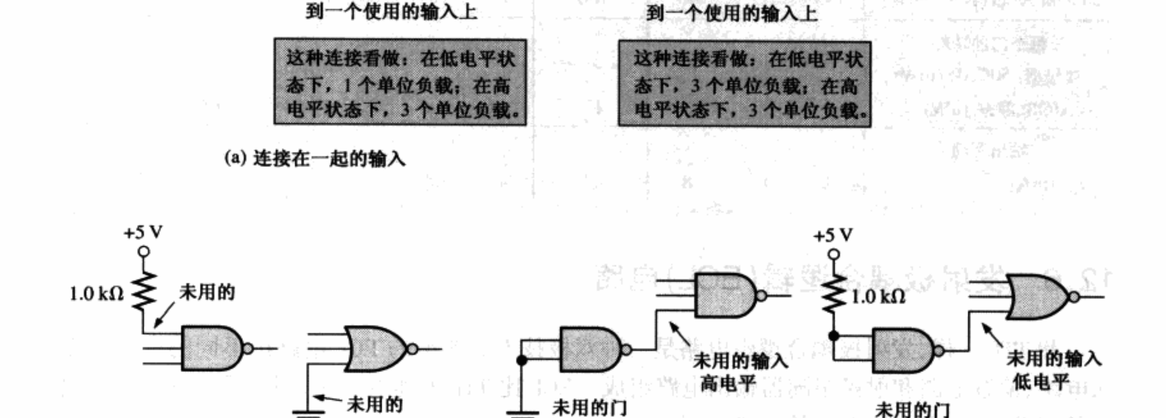


图 12.45　未用的 TTL 输入的处理方法

输入连接到 V_{CC}或接地　可以把与门和与非门的未用输入通过一个 1.0 kΩ 的电阻连接到 V_{CC}。这样连接可使未用的输入拉到高电平。或门和或非门的未用输入可以连接到地。这些方法如图 12.45(b)所示。

12.5　CMOS 和 TTL 的性能比较

在这一节,我们将对 CMOS 系列主要的工作和性能特点与主要的 TTL 系列及 BiCMOS 进行比较。

学完本节以后,应该能够

- 在传输延迟、最大时钟频率、功耗和驱动能力方面,比较 TTL(双极型)、BiMOS 和 CMOS 芯片

过去,与 CMOS 电路相比,TTL(双极)的出众特性表现在它相对较高的速度和输出电流能力方面。今天,正如所知,虽然 TTL 仍然还在使用,但是 CMOS 在许多领域通常与 TTL 相当或比 TTL 更为出众,在这一点上使得 TTL 已经减少,而 CMOS 已成为了主流的 IC 技术。IC 逻辑芯片中有一个系列 BiCMOS,它使用了 CMOS 逻辑与 TTL 输出的电路,并把两者优点结合起来。表 12.1 提供了几种 IC 系列的性能参数比较。

表 12.1 对几种 74××IC 系列选择的性能参数进行了比较

	双极型(TTL)			BiCMOS	CMOS					
					5 V			3.3 V		
	F	LS	ALS	ABT	HC	AC	AHC	LV	LVC	ALVC
速度										
门传输延迟, t_p (ns)	3.3	10	7	3.2	7	5	3.7	9	4.3	3
FF 最大时钟频率(MHz)	145	33	45	150	50	160	170	90	100	150
每个门的功耗										
双极型: 50% dc (mW)	6	2.2	1.4							
CMOS: 静态 (μW)				17	2.75	0.55	2.75	1.6	0.8	0.8
输出驱动										
I_{OL} (mA)	20	8	8	64	4	24	8	12	24	24

12.6 发射极耦合逻辑(ECL)电路

和 TTL 一样,发射极耦合逻辑电路是一种双极技术。典型的 ECL 电路由不同的放大器输入电路、偏置电路和射极跟随器输出电路组成。ECL 比 TTL 的速度更快,因为晶体管不会在饱和状态下工作,它用于专门的高速应用中。

学完本节以后,应当能够

- 描述 ECL 与 TTL 和 CMOS 的区别
- 解释 ECL 的优缺点

ECL 或/或非门电路如图 12.46(a)所示。射极跟随器输出电路提供了或逻辑功能和或非的补码,如图 12.46(b)所示。

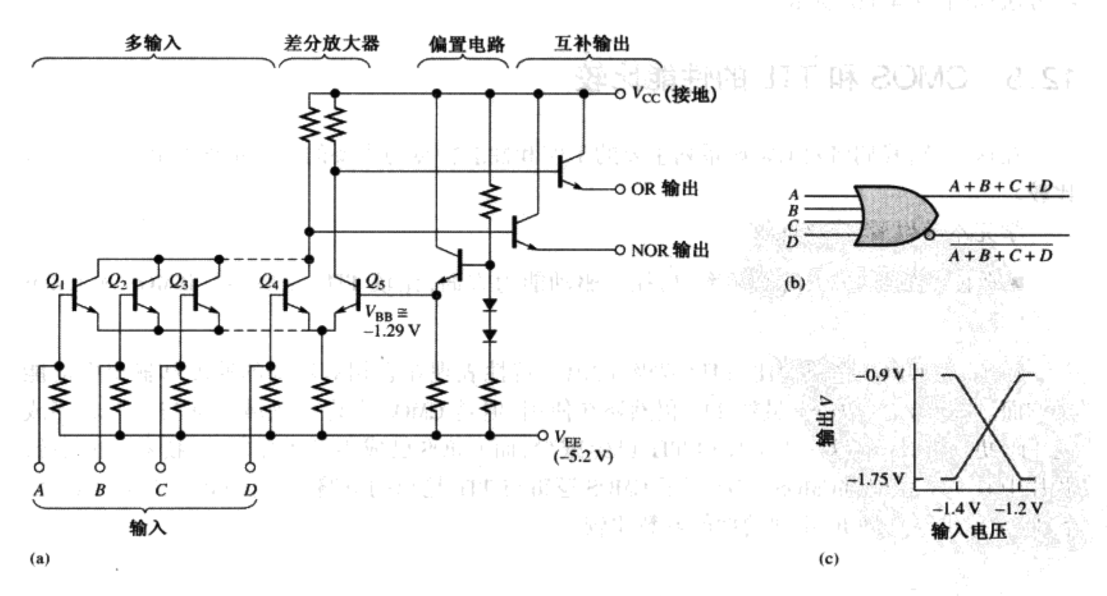


图 12.46 一个 ECL OR/NOR 门电路

因为发射极跟随器的输出阻抗较低，而差分放大器输入的输入阻抗较高，所以高扇出的驱动是可能的。在这种类型的电路中，不可能出现饱和状态。由于没有饱和状态，导致较高的功耗和受到限制的电压波动(小于 1 V)，但允许在高频开关下工作。

V_{CC}引脚通常是连接到地，V_{EE}引脚连接电源电压 -5.2 V，以便处于最佳工作状态。注意图 12.46(c)中输出对地电平从低电平 -1.75 V 到高电平 -0.9 V 的变化。在正逻辑中，1 是高电平(负值，值较小)，0 是低电平(负值，值较大)。

12.6.1 噪声容限

正如我们所学到的，门电路的噪声容限是衡量对不需要的电压波动(噪声)的抗干扰能力。典型的 ECL 电路的噪声容限大约从 0.2 V 到 0.25 V。这两个数字都小于 TTL 的指标，从而使得 ECL 不太适合噪声大的环境。

12.6.2 ECL 与 TTL 和 CMOS 的比较

表 12.2 给出了对 F、AHC 和 ECL 的关键性能参数的比较。

表 12.2 ECL 系列的特性参数与 F 和 AHC 的特性参数的比较

	双极型 (TTL) F	CMOS AHC	双极型 (ECL)
速度			
门传输延迟，t_p(ns)	3.3	3.7	0.22~1
FF 最大时钟频率 (MHz)	145	170	330~2800
每个门的功耗			
双极型：50% dc	8.9 mW		25 mW~73 mW
CMOS：静态		2.5 μW	

12.7 PMOS、NMOS 和 E^2CMOS

PMOS 和 NMOS 电路在 LSI 中大量使用，如在长移位寄存器、大容量存储器和微处理器产品中。这样的用法是 MOS 晶体管的优点导致的结果，即低功耗和所需芯片面积很小。E^2CMOS 用于可重复编程的 PLD 中。

学完本节后，应当能够

- 描述基本的 PMOS 门电路
- 描述基本的 NMOS 门电路
- 描述基本的 E^2CMOS 单元

12.7.1 PMOS

最初产生的高密度 MOS 电路技术之一就是 PMOS。它使用增强模式的 p 型沟道 MOS 晶体管构成了基本的门电路构建模块。图 12.47 给出了产生正逻辑或非功能的基本 PMOS 门电路。

PMOS 门电路的工作原理如下：电源电压 V_{GG}是一个负电压，V_{CC}是一个正电压或地(0 V)。晶体管 Q_3 是一直处于偏置状态，以生成一个恒定的漏源电阻。它的唯一目的就是作为限流电阻。如果一个高电平(V_{CC})加在输入 A 或输入 B 上，那么 Q_1 或 Q_2 截止，输出被拉到一个接

近 V_{GG}的电压,表示低电平。如果一个低电平(V_{GG})加在输入 A 和输入 B 上,那么 Q_1 和 Q_2 导通。造成输出变为高电平(接近 V_{CC})。因为两个输入中只要有一个是高电平时,输出就是低电平,而只有所有的输入都是低电平时,输出才是高电平,所以得到了一个或非门。

12.7.2 NMOS

NMOS 芯片是作为改进的处理技术开发的。NMOS 电路使用 n 型沟道 MOS 晶体管,如图 12.48 中的与非门和或非门电路所示。

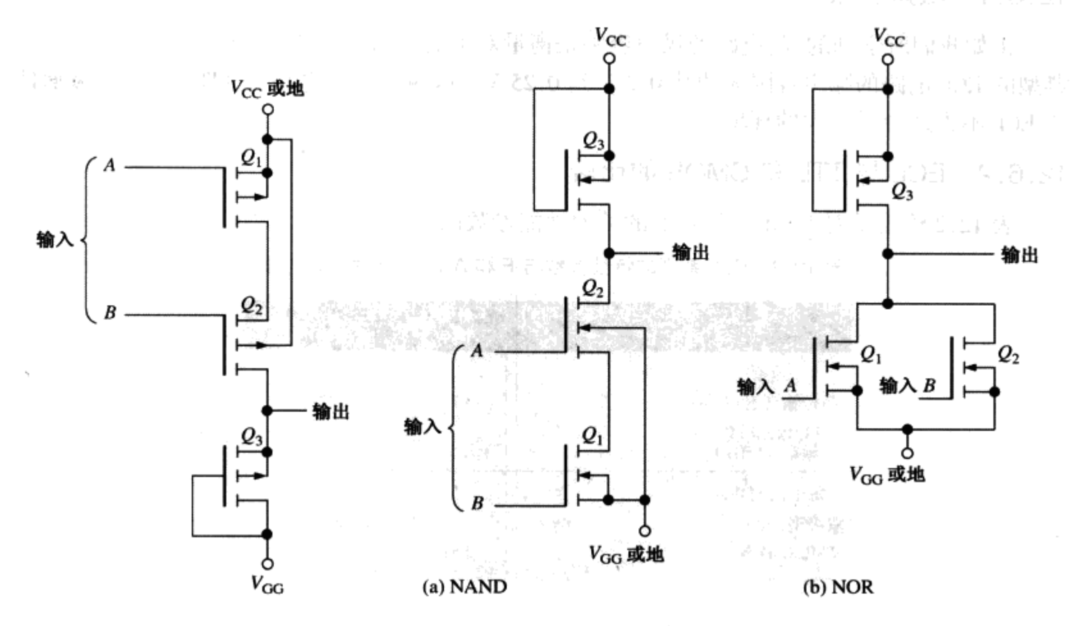


图 12.47　基本 PMOS 门　　　　图 12.48　两个 NMOS 门

在图 12.48(a)中,Q_3 作为一个限流电阻。当一个低电平(V_{GG}或地)加到一个输入或两个输入上时,那么至少有一个晶体管(Q_1 或 Q_2)截止,输出被拉到高电平,接近 V_{CC}。当高电平加到输入 A 和输入 B 上时,Q_1 和 Q_2 都导通,输出为低电平。当然,这个分析结果得出此电路是一个或非门电路。

在图 12.48(b)中,Q_3 再次作为一个电阻。两个输入中有一个是高电平,就会使 Q_1 或 Q_2 导通,并将输出拉到低电平。当两个输入都是低电平时,两个晶体管都截止,输出被拉到高电平。

12.7.3 E^2CMOS

E^2CMOS(电可擦除)技术基于 CMOS 和 NMOS 技术的结合,用于可编程逻辑芯片中,例如 PROM 和 CPLD。一个 E^2CMOS 单元是围绕带浮动门的 MOS 晶体管构建的,这种浮动门由一个不大的编程电流进行外部充电和放电。这种类型的单元示意图如图 12.49 所示。

当浮动门通过移动电子充电到正电压时,读出晶体管导通,并存储二进制 0。当浮动门电路通过放入电子充电到负电压时,读出晶体管截止,并存储二进制 1。控制门控制浮动门的电压。在使用字和位线的读写操作过程中,传递晶体管把读出晶体管与字和位线阵列隔离开来。

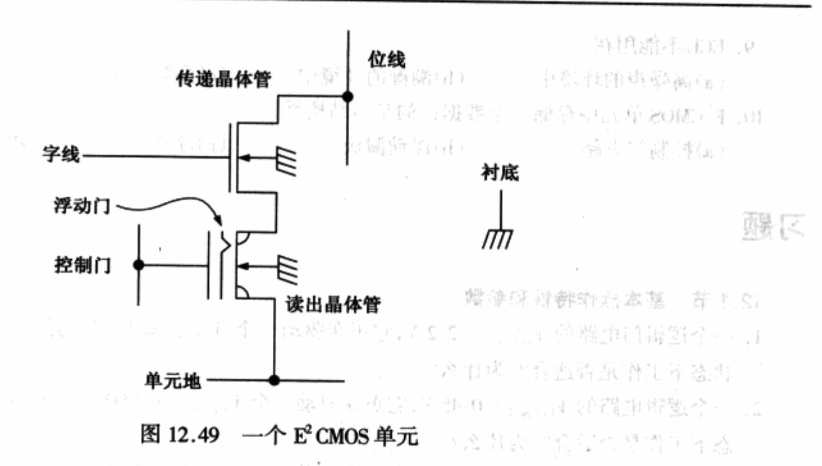
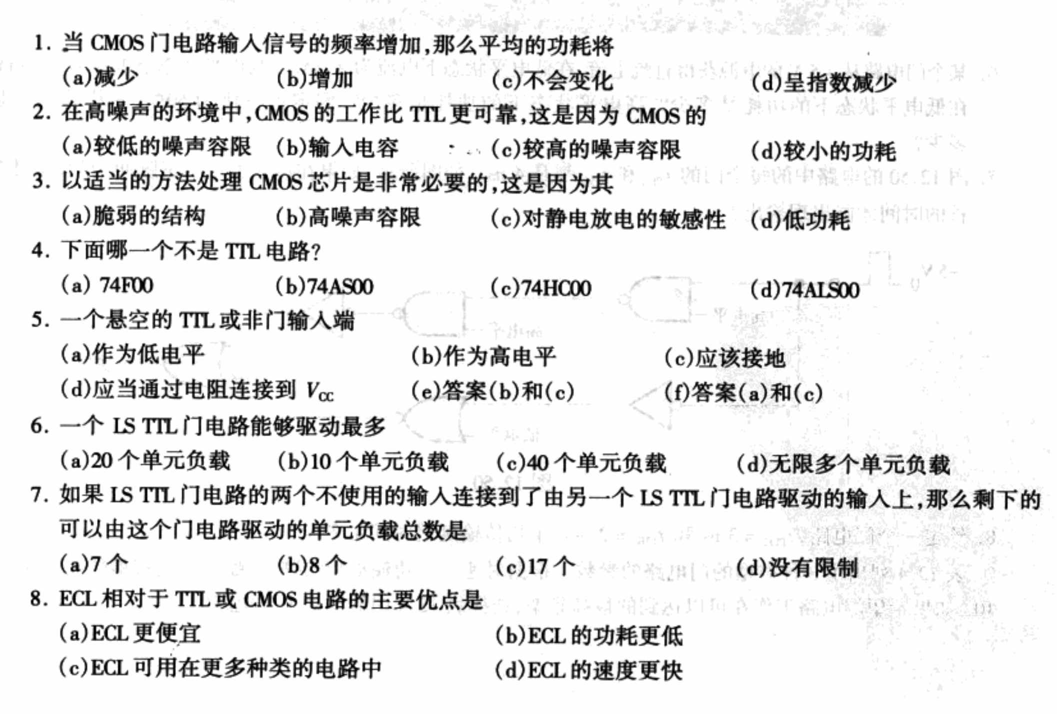


图 12.49　一个 E^2CMOS 单元

E^2CMOS 单元可通过在控制门或单元的位线加一个编程脉冲进行编程，而位线已由字线上的电压选中。在编程周期中，首先通过在控制门上加一个电压使得浮动门为负电压，从而使 E^2CMOS 单元的内容被擦除。这将使读出晶体管处于截止状态(存储一个 1)。在单元的位线加一个写脉冲将保存一个 0。这将使浮动门充电到一个使读出晶体管导通(存储 0)的电压，这个电压在这个点检测晶体管处于打开状态(存储一个 0)。通过读取位线上是否存在一个小的单元电流，单元中存储的位就被读出。如果存储的是 1，那么就没有单元电流，因为读出晶体管截止。如果存储的是 0，则有一个小的单元电流，因为读出晶体管导通。一旦在单元中存储了一位，就会永远被保存，除非擦除这个单元或者在这个单元中写入了一个新的位。

自测题　(答案在本章的结尾)



1. 当 CMOS 门电路输入信号的频率增加，那么平均的功耗将
   (a)减少　(b)增加　(c)不会变化　(d)呈指数减少
2. 在高噪声的环境中，CMOS 的工作比 TTL 更可靠，这是因为 CMOS 的
   (a)较低的噪声容限　(b)输入电容　(c)较高的噪声容限　(d)较小的功耗
3. 以适当的方法处理 CMOS 芯片是非常必要的，这是因为其
   (a)脆弱的结构　(b)高噪声容限　(c)对静电放电的敏感性　(d)低功耗
4. 下面哪一个不是 TTL 电路?
   (a) 74F00　(b)74AS00　(c)74HC00　(d)74ALS00
5. 一个悬空的 TTL 或非门输入端
   (a)作为低电平　(b)作为高电平　(c)应该接地
   (d)应当通过电阻连接到 V_{CC}　(e)答案(b)和(c)　(f)答案(a)和(c)
6. 一个 LS TTL 门电路能够驱动最多
   (a)20 个单元负载　(b)10 个单元负载　(c)40 个单元负载　(d)无限多个单元负载
7. 如果 LS TTL 门电路的两个不使用的输入连接到了由另一个 LS TTL 门电路驱动的输入上，那么剩下的可以由这个门电路驱动的单元负载总数是
   (a)7 个　(b)8 个　(c)17 个　(d)没有限制
8. ECL 相对于 TTL 或 CMOS 电路的主要优点是
   (a)ECL 更便宜　(b)ECL 的功耗更低
   (c)ECL 可用在更多种类的电路中　(d)ECL 的速度更快

9. ECL不能用在
   (a)高噪声的环境中　(b)潮湿的环境中　(c)高频应用中
10. E^2CMOS单元中存储一个数据位的基本结构是
   (a)控制门电路　(b)浮动漏极　(c)浮动门　(d)单元电流

习题

12.1节　基本操作特性和参数

1. 一个逻辑门电路的 $V_{OH(min)} = 2.2$ V,它正在驱动一个 $V_{IH(min)} = 2.5$ V的门电路。这两个电路在高电平状态下工作是否适合?为什么?
2. 一个逻辑电路的 $V_{OL(max)} = 0.45$ V,它正在驱动一个 $V_{IL(max)} = 0.75$ V的门电路。这两个电路在低电平状态下工作是否适合?为什么?
3. 一个TTL门电路具有下列实际的电压电平值: $V_{IH(min)} = 2.25$ V, $V_{IL(max)} = 0.65$ V。假设它正在由一个 $V_{OH(min)} = 2.4$ V和 $V_{OL(max)} = 0.4$ V的门电路驱动。那么高电平和低电平状态下的噪声容限是多少?
4. 对于习题3中的门电路,在高电平和低电平状态下,它所能允许的输入上的噪声尖峰电压的最大振幅是多少?
5. 表12.3中给出了3种类型的逻辑门的电压规范。请选择可在高噪声工业环境中使用的门电路。

表 12.3

	$V_{OH(MIN)}$	$V_{OL(MAX)}$	$V_{IH(MIN)}$	$V_{IL(MAX)}$
门 A	2.4 V	0.4 V	2 V	0.8 V
门 B	3.5 V	0.2 V	2.5 V	0.6 V
门 C	4.2 V	0.2 V	3.2 V	0.8 V

6. 某个门电路从+5 V的电源获得直流电流,在低电平状态下电流为2 mA,在高电平状态下电流为3.5 mA。在低电平状态下的功耗是多少?高电平状态下的功耗是多少?假定占空比为50%,平均功耗又是多少?
7. 图12.50的电路中的每个门的 t_{PLH} 和 t_{PHL} 都是4 ns。如图所示,如果在输入加一个正脉冲,那么经过多长的时间才能出现输出?

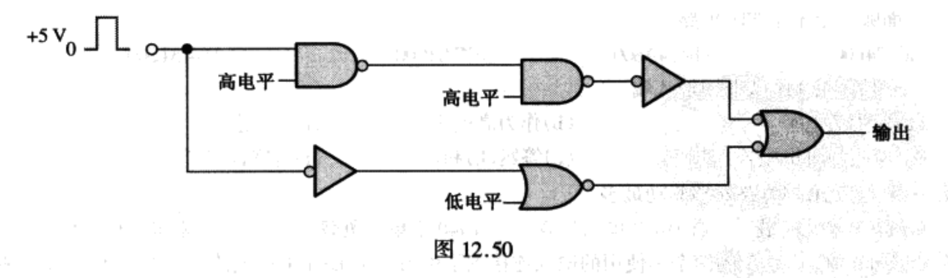


图 12.50

8. 给定一个门电路, $t_{PLH} = 3$ ns 和 $t_{PHL} = 2$ ns。平均传输延迟时间是多少?
9. 表12.4列出了3种类型的门电路的参数。根据对速率-功耗乘积的判断,哪一个具有最佳的性能?
10. 如果希望门电路工作在可以达到的最高频率,选择表12.4中的哪一个门电路?

表 12.4

	t_{PLH}	t_{PHL}	P_D
门*A*	1 ns	1.2 ns	15 mW
门*B*	5 ns	4 ns	8 mW
门*C*	10 ns	10 ns	0.5 mW

11. 标准的 TTL 门电路的扇出系数是 10。图 12.51 中是否有门电路超载了？如果有，是哪一个？

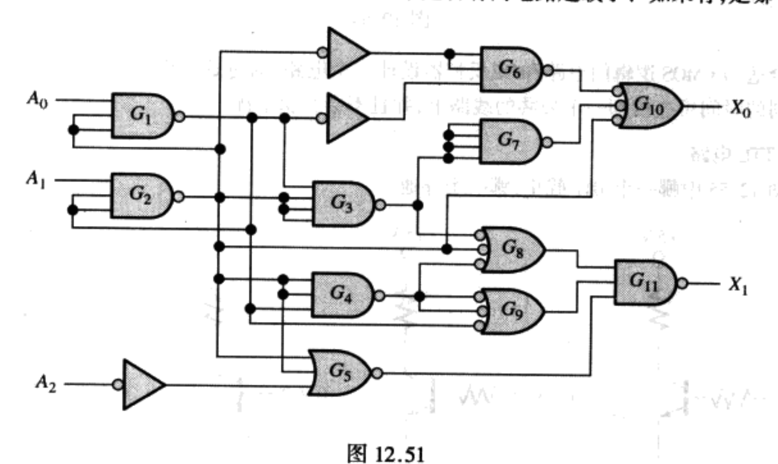


图 12.51

12. 图 12.52 中的哪一个 CMOS 门电路网路的可以在最高频率下工作？

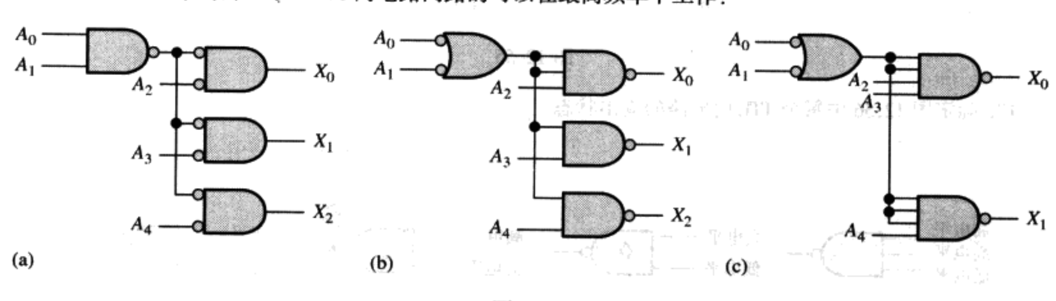


图 12.52

12.2 节 CMOS 电路

13. 确定图 12.53 中每一个 MOS 场效应管的状态(导通或截止)。

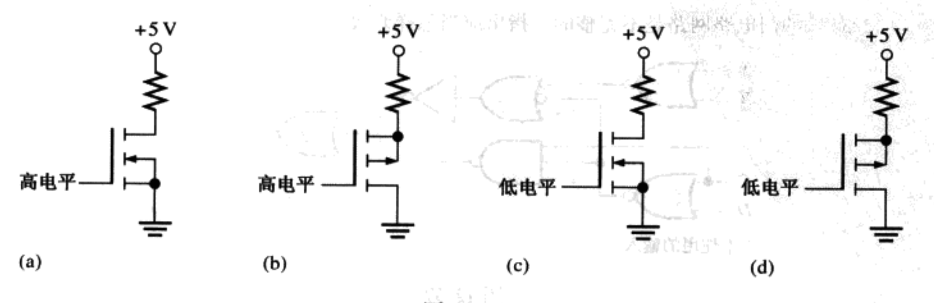


图 12.53

14. 图 12.54 中的 CMOS 门电路网路是不完整的。指出应当怎样修改。

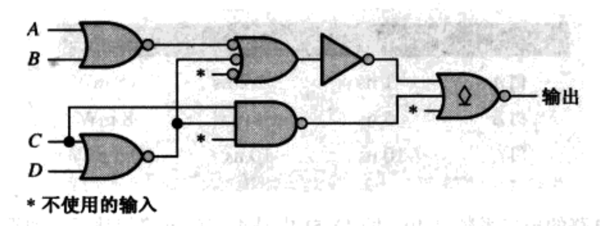


图 12.54

15. 使用合适的 CMOS 逻辑门电路和/或反相器设计一个电路,通过这个电路,来自 4 个不同源头的信号在不同的时间可连接到一个公共的线路上,并且不会互相干扰。

12.3 节 TTL 电路

16. 确定图 12.55 中哪一个 BJT 截止,哪一个导通。

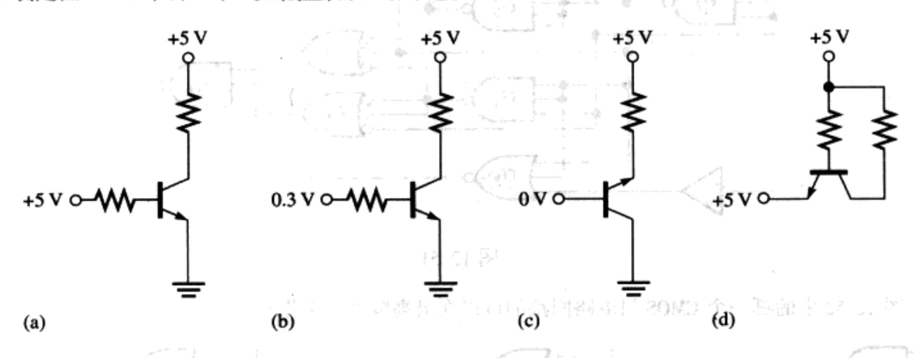


图 12.55

17. 确定图 12.56 中每个 TTL 门电路的输出状态。

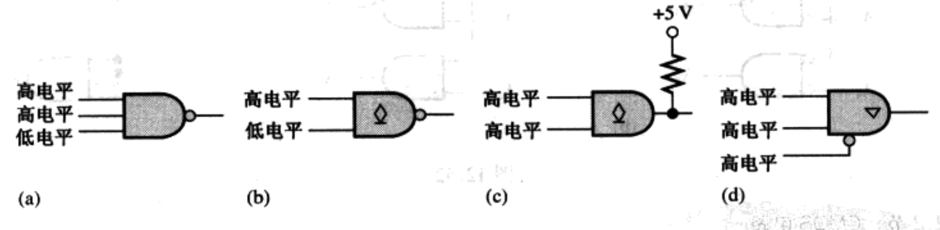


图 12.56

18. 图 12.57 中的门电路网路是不完整的。指出应当怎样修改。

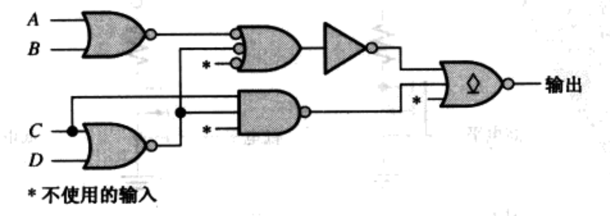


图 12.57

12.4 节　TTL 在实际使用中的注意事项

19. 确定图 12.58 中每个 TTL 门电路的输出电平。

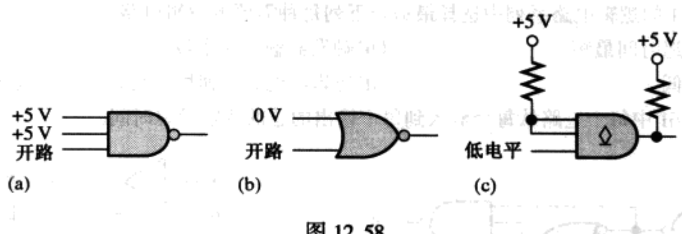


图 12.58

20. 针对图 12.59 中的每个部分，说出每一个驱动门电路是灌电流还是拉电流。请详细说明每种情况下驱动门电路流进或流出的最大电流。所有的门电路都是标准 TTL。

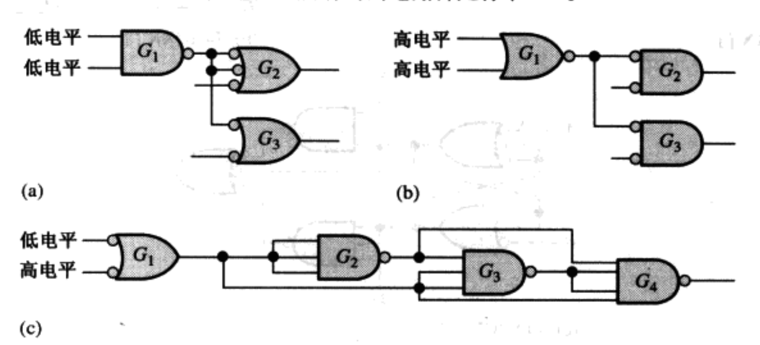


图 12.59

21. 使用集电极开路反相器实现下列逻辑表达式：
    (a) $X=\overline{A}\overline{B}\overline{C}$　　(b) $X=A\overline{B}C\overline{D}$　　(c) $X=ABC\overline{D}\overline{E}\overline{F}$
22. 写出图 12.60 中每个电路的逻辑表达式。

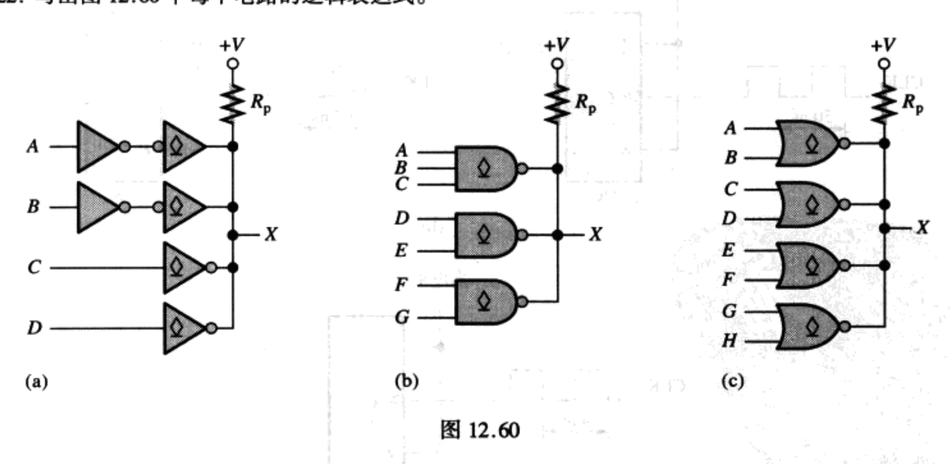


图 12.60

23. 如果对每个门电路 $I_{OL(max)}=40$ mA 和 $V_{OL(max)}=0.25$ V，确定图 12.60 中每个上拉电阻的最小值。假设输出 X 驱动了 10 个标准 TTL 单位负载，电源电压是 5 V。
24. 一个给定的继电器需要 60 mA 的电流。请使用集电极开路与非门电路驱动这个中继器，集电极开路与非门的 $I_{OL(max)}=40$ mA。

12.5节 CMOS和TTL的性能比较

25. 请在表12.1中选出速率－功耗乘积最好的IC系列。

26. 请在表12.1的逻辑电路系列中选择最适合下列每种需要的逻辑电路：

(a)传输延迟时间最短 (b)触发器触发速率最快

(c)功耗最低 (d)逻辑门电路的速度和功耗之间的最佳折中

27. 确定图12.61中每个电路从每个输入到每个输出的总的传输延迟时间。

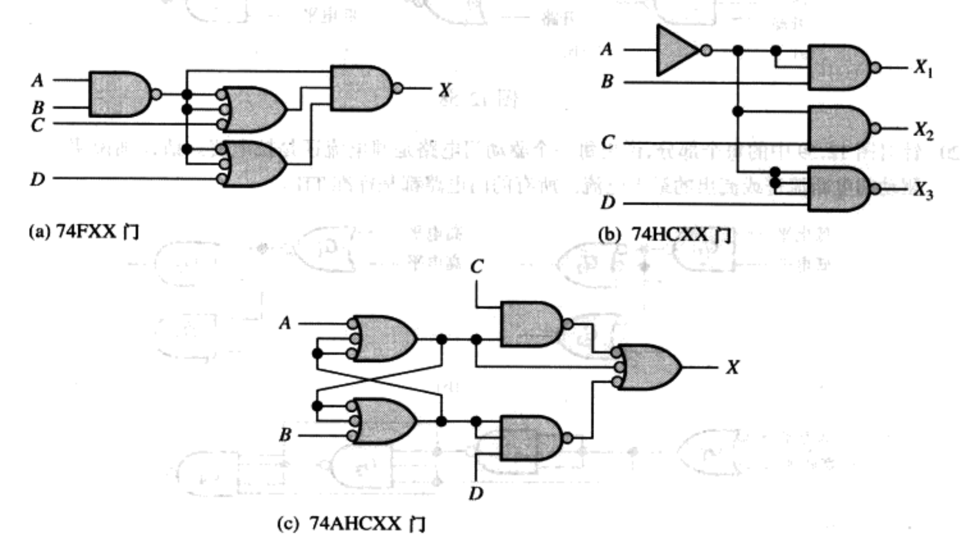


图12.61

28. 图12.62中的触发器中有一个可能有不稳定的输出。如果有，是哪一个，为什么？

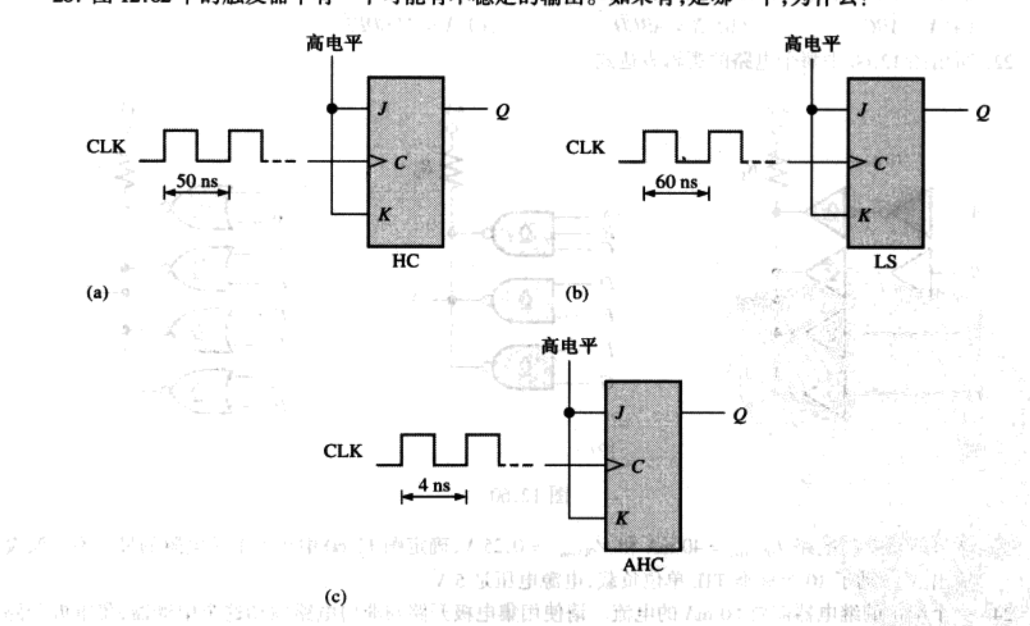


图12.62

12.6 节　发射极耦合逻辑(ECL)电路

29. ECL 逻辑电路系统和 TTL 逻辑电路系统之间有哪些基本的差别?

30. 针对下列每一种需求,选择 ECL、HCMOS 或合适的 TTL 系列:

(a)速度最快　(b)功耗最低

(c)高速度和低功耗之间(速率－功耗乘积)的最佳折中

自测题答案

1. (b)　2. (c)　3. (c)　4. (c)　5. (e)　6. (a)　7. (c)　8. (d)　9. (a)　10. (c)

奇数题目的答案

第 1 章　数字概念

1. 数字信号可以更有效和更可靠地传输和存储。
3. (a) 11010001　(b) 000101010
5. (a) 550 ns　(b) 600 ns　(c) 2.7 μs　(d) 10 V
7. 250 Hz
9. 50%
11. 8 μs;1 μs

第 2 章　数字系统、运算和编码

1. (a) 1　(b) 100　(c) 100 000
3. (a) 400;70;1　(b) 9000;300;50;6　(c) 100 000;20 000;5000;0;0;0
5. (a) 3　(b) 4　(c) 7　(d) 8　(e) 9　(f) 12　(g) 11　(h) 15
7. (a) 51.75　(b) 42.25　(c) 65.875　(d) 120.625　(e) 92.656 25　(f) 113.0625　(g) 90.625　(h) 127.968 75
9. (a) 5 比特　(b) 6 比特　(c) 6 比特　(d) 7 比特　(e) 7 比特　(f) 7 比特　(g) 8 比特　(h) 8 比特
11. (a) 1010　(b) 10001　(c) 11000　(d) 110000　(e) 111101　(f) 1011101　(g) 1111101　(h) 10111010
13. (a) 1111　(b) 10101　(c) 11100　(d) 100010　(e) 101000　(f) 111011　(g) 1000001　(h) 1001001
15. (a) 100　(b) 100　(c) 1000　(d) 1101　(e) 1110　(f) 11000
17. (a) 1001　(b) 1000　(c) 100011　(d) 110110　(e) 10101001　(f) 10110110
19. (a) 010　(b) 001　(c) 0101　(d) 00101000　(e) 0001010　(f) 11110
21. (a) 00011101　(b) 11010101　(c) 01100100　(d) 11111011
23. (a) 00001100　(b) 10111100　(c) 01100101　(d) 10000011
25. (a) −102　(b) +116　(c) −64
27. (a) 0 10001101 11110000101011000000000　(b) 1 10001010 11000001100000000000000
29. (a) 00110000　(b) 00011101　(c) 11101011　(d) 100111110
31. (a) 11000101　(b) 11000000
33. 100111001010
35. (a) 00111000　(b) 01011001　(c) 101000010100　(d) 010111001000　(e) 0100000100000000　(f) 1111101100010111　(g) 1000101010011101
37. (a) 35　(b) 146　(c) 26　(d) 141　(e) 243　(f) 235　(g) 1474　(h) 1792
39. (a) 60_{16}　(b) $10B_{16}$　(c) $1BA_{16}$
41. (a) 10　(b) 23　(c) 46　(d) 52　(e) 67　(f) 367　(g) 115　(h) 532　(i) 4085
43. (a) 001011　(b) 101111　(c) 001000001　(d) 011010001　(e) 101100000　(f) 100110101011　(g) 001011010111001　(h) 100101110000000　(i) 001000000010001011
45. (a) 00010000　(b) 00010011　(c) 00011000　(d) 00100001　(e) 00100101　(f) 00110110　(g) 01000100　(h) 01010111　(i) 01101001　(j) 10011000　(k) 000100100101　(l) 000101010110

47. (a) 000100000100 (b) 000100101000 (c) 000100110010 (d) 000101010000 (e) 000110000110 (f) 001000010000 (g) 001101011001 (h) 010101000111 (i) 0001000001010001
49. (a) 80 (b) 237 (c) 346 (d) 421 (e) 754 (f) 800 (g) 978 (h) 1683 (i)9018 (j) 6667
51. (a) 00010100 (b) 00010010 (c) 00010111 (d) 00010110 (e) 01010010 (f) 000100001001 (g) 000110010101 (h) 0001001001101001
53. 当格雷码在序列中每次从一个数变到下一个数时，格雷码仅有一位发生变化。
55. (a) 1100 (b) 00011 (c) 10000011110
57. (a) CAN (b) J (c) = (d) # (e) > (f) B
59. 48 65 6C 6C 6F 2E 20 48 6F 77 20 61 72 65 20 79 6F 75 3F
61. (b) 不正确
63. (a) 110100100 (b) 000001001 (c) 111111110
65. 001010001
67. (a) 110100010 (b) 100000101

第3章 逻辑门

1. 参见图 P.1

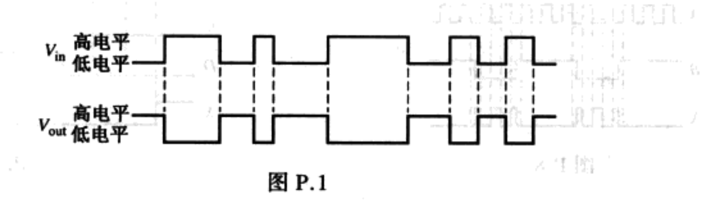


图 P.1

3. 参见图 P.2
5. 参见图 P.3

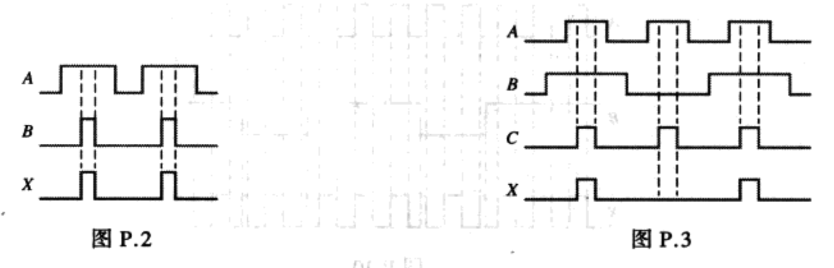


图 P.2 图 P.3

7. 参见图 P.4
9. 参见图 P.5

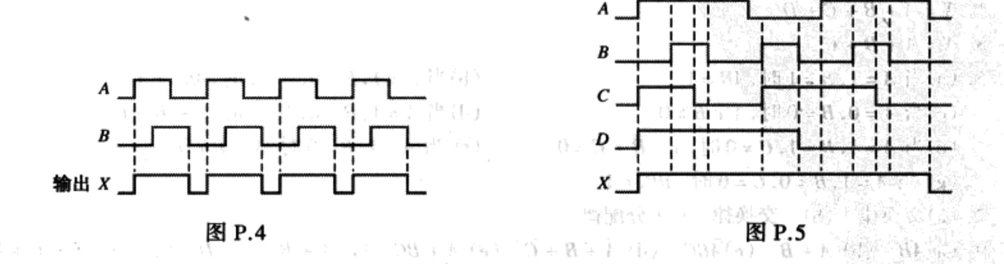


图 P.4 图 P.5

11. 参见图 P.6

13. 参见图 P.7

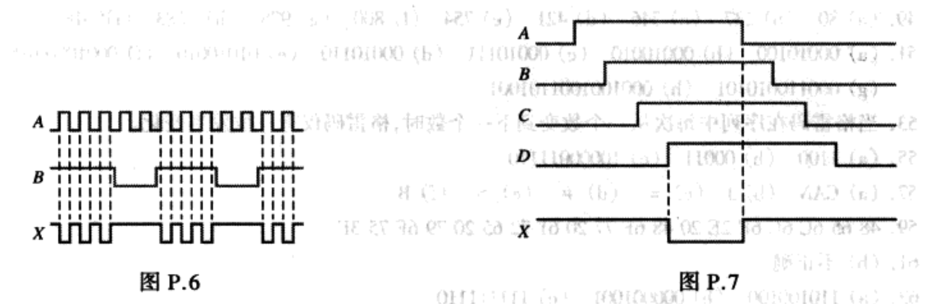


图 P.6　　图 P.7

15. 参见图 P.8

17. 参见图 P.9

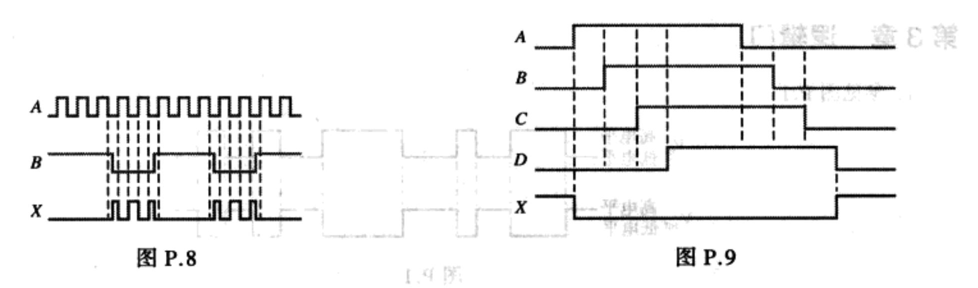


图 P.8　　图 P.9

19. XOR = $A\bar{B} + \bar{A}B$；OR = $A + B$

21. 参见图 P.10

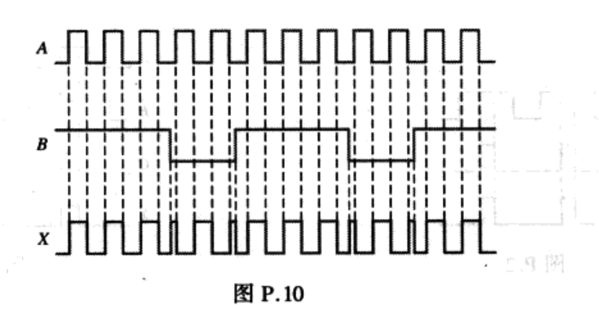


图 P.10

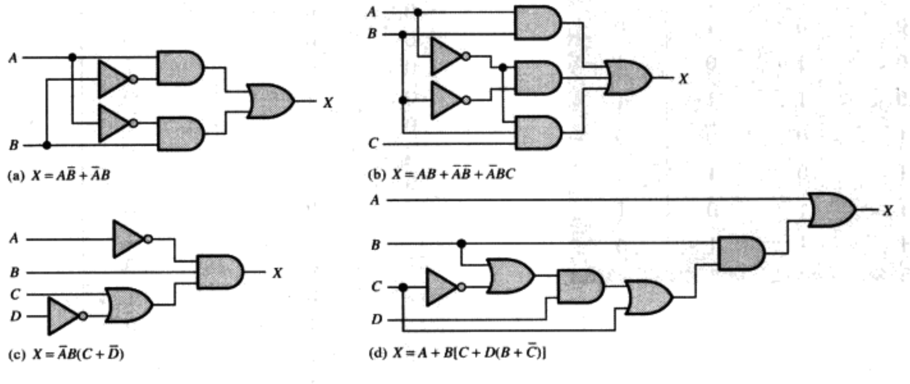
第 4 章　布尔代数和逻辑化简

1. $X = A + \bar{B} + C + D$
3. $X = \bar{A} + \bar{B} + \bar{C}$
5. (a)当 $A = 1, B = 1$ 时，$AB = 1$　　(b)当 $A = 1, B = 0, C = 1$ 时，$A\bar{B}C = 1$
   (c)当 $A = 0, B = 0$ 时，$A + B = 0$　　(d)当 $A = 1, B = 0, C = 1$ 时，$\bar{A} + B + C = 0$
   (e)当 $A = 1, B = 1, C = 0$ 时，$\bar{A} + \bar{B} + C = 0$　　(f) 当 $A = 1, B = 0$ 时，$\bar{A} + B = 0$
   (g) 当 $A = 1, B = 0, C = 0$ 时，$A\bar{B}\bar{C} = 1$
7. (a)交换律　(b)　交换律　(c) 分配律
9. (a)$\bar{A}B$　(b) $A + \bar{B}$　(c)$\bar{A}\bar{B}\bar{C}$　(d) $\bar{A} + \bar{B} + \bar{C}$　(e) $\bar{A} + \bar{B}\bar{C}$　(f) $\bar{A} + \bar{B} + \bar{C} + \bar{D}$　(g) $(\bar{A} + \bar{B})(\bar{C} + \bar{D})$
   (h) $\bar{A}B + C\bar{D}$

11. (a) $(\bar{A}+\bar{B}+\bar{C})(\bar{E}+\bar{F}+\bar{G})(\bar{H}+\bar{I}+\bar{J})(\bar{K}+\bar{L}+\bar{M})$ (b) $\bar{A}B\bar{C}+BC$ (c) $\overline{ABCDEFGH}$

13. (a) $X=ABCD$ (b) $X=AB+C$ (c) $X=\overline{\bar{A}B}$ (d) $X=(A+B)C$

15. 参见图 P.11



(a) $X=A\bar{B}+\bar{A}B$ (b) $X=AB+\bar{A}\bar{B}+\bar{A}BC$

(c) $X=\bar{A}B(C+\bar{D})$ (d) $X=A+B[C+D(B+\bar{C})]$

图 P.11

17. (a) A (b) AB (c) C (d) A (e) $\bar{A}C+\bar{B}C$

19. (a) $BD+BE+\bar{D}F$ (b) $\bar{A}\bar{B}C+\bar{A}\bar{B}D$ (c) B (d) $AB+CD$ (e) ABC

21. (a) $A\bar{B}+AC+BC$ (b) $AC+\bar{B}C$ (c) $AB+AC$

23. (a)变量域:A,B,C 最小项: $A\bar{B}C+A\bar{B}\bar{C}+ABC+\bar{A}BC$

(b)变量域:A,B,C 最小项: $ABC+A\bar{B}C+\bar{A}\bar{B}C$

(c)变量域:A,B,C 最小项:$ABC+AB\bar{C}+A\bar{B}C$

25. (a) 101 + 100 + 111 + 011 (b) 111 + 101 + 001 (c) 111 + 110 + 101

27. (a) $(A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+\bar{B}+C)$

(b) $(A+B+C)(A+\bar{B}+C)(A+\bar{B}+\bar{C})(\bar{A}+B+C)(\bar{A}+\bar{B}+C)$

(c) $(A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(A+\bar{B}+\bar{C})(\bar{A}+B+C)$

29. (a) 参见表 P.1

29. (b) 参见表 P.2

表 P.1

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

表 P.2

X	Y	Z	Q
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

31. (a)参见表 P.3

31. (b)参见表 P.4

表 P.3

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

表 P.4

W	X	Y	Z	Q
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

33.(a)参见表 P.5

33.(b)参见表 P.6

表 P.5

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

表 P.6

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

35. 参见图 P.12

37. 参见图 P.13

39. (a)没有简化　(b)AC　(c) $\overline{D}\overline{F} + E\overline{F}$

41. (a)$AB + AC$　(b)$A + BC$　(c)$B\overline{C}D + A\overline{C}D + BC\overline{D} + AC\overline{D}$　(d)$A\overline{B} + CD$

43. $\overline{B} + C$

45. $\overline{A}\overline{B}\overline{C}\overline{D} + C\overline{D} + BC + A\overline{D}$

47. (a)$(A + \overline{B} + C + \overline{D})(\overline{A} + B + \overline{C} + D)(\overline{A} + \overline{B} + \overline{C} + \overline{D})$　(b) $(W + \overline{Z})(W + X)(\overline{Y} + \overline{Z})(X + \overline{Y})$

49. $(A+C+D)(A+\bar{B}+C)(\bar{A}+B+\bar{D})(B+\bar{C}+\bar{D})(\bar{A}+\bar{B}+\bar{C}+D)$

51. LED。LED 发射光，LCD 不发射。

53. 少一个反相器和六个门。

AB \ C	0	1
00	000	001
01	010	011
11	110	111
10	100	101

图 P.12

AB \ C	0	1
00	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$
01	$\bar{A}B\bar{C}$	$\bar{A}BC$
11	$AB\bar{C}$	ABC
10	$A\bar{B}\bar{C}$	$A\bar{B}C$

图 P.13

第 5 章 组合逻辑

1. 参见图 P.14

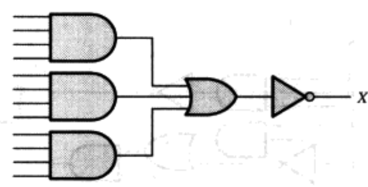


图 P.14

3. (a) $X = ABB$
   (b) $X = AB + B$
   (c) $X = \bar{A} + B$
   (d) $X = (A + B) + AB$
   (e) $X = \overline{\bar{A}BC}$
   (f) $X = (A + B)(\bar{B} + C)$

5. (a)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(b)

A	B	X
0	0	0
0	1	1
1	0	0
1	1	1

(c)

A	B	X
0	0	1
0	1	1
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

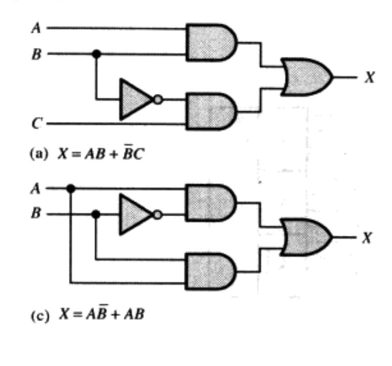
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(f)

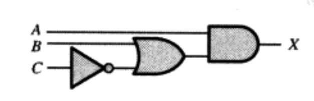
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

7. $X = \overline{A\bar{B} + \bar{A}B} = (\bar{A} + B)(A + \bar{B})$

9. 参见图 P.15

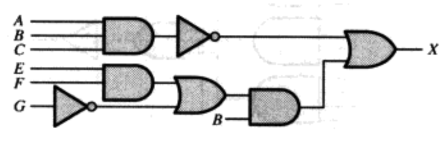


(a) $X = AB + \bar{B}C$

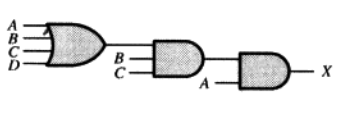


(b) $X = A(B + \bar{C})$

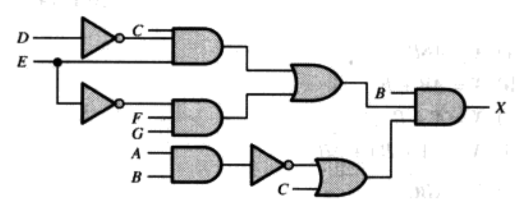
(c) $X = A\bar{B} + AB$



(d) $X = \overline{ABC} + B(EF + \bar{G})$



(e) $X = A[BC(A + B + C + D)]$



(f) $X = B(C\bar{D}E + \bar{E}FG)(\overline{AB} + C)$

图 P.15

11. 参见图 P.16

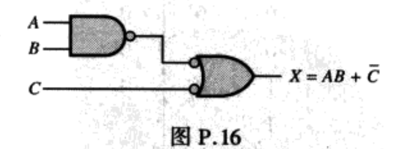


图 P.16

13. $X = AB$

15. (a) 没有简化 (b) 没有简化

(c) $X = A$ (d) $X = \bar{A} + \bar{B} + \bar{C} + EF + \bar{G}$

(e) $X = ABC$

(f) $X = BC\bar{D}E + \bar{A}B\bar{E}FG + BC\bar{E}FG$

17. (a) $X = AC + AD + BC + BD$ (b) $X = \bar{A}CD + \bar{B}CD$

(c) $X = ABD + CD + E$ (d) $X = \bar{A} + B + D$

(e) $X = ABD + \bar{C}D + \bar{E}$

(f) $X = \bar{A}\bar{C} + \bar{A}\bar{D} + \bar{B}\bar{C} + \bar{B}\bar{D} + \bar{E}\bar{G} + \bar{E}\bar{H} + \bar{F}\bar{G} + \bar{F}\bar{H}$

19. 参见图 P.17

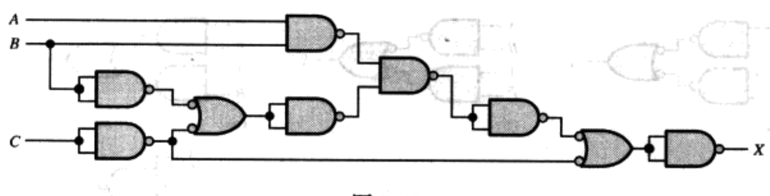


图 P.17

21. 参见图 P.18

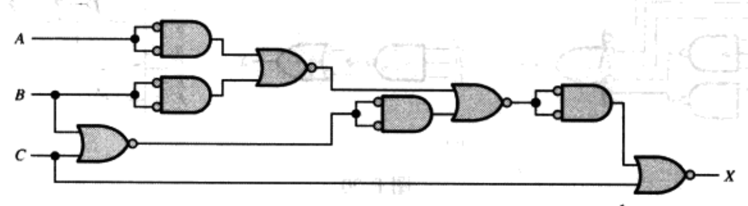
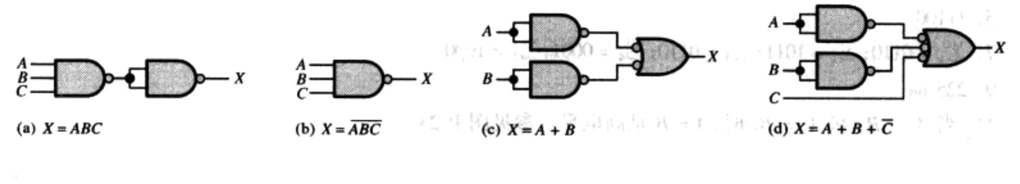


图 P.18

23. 参见图 P.19



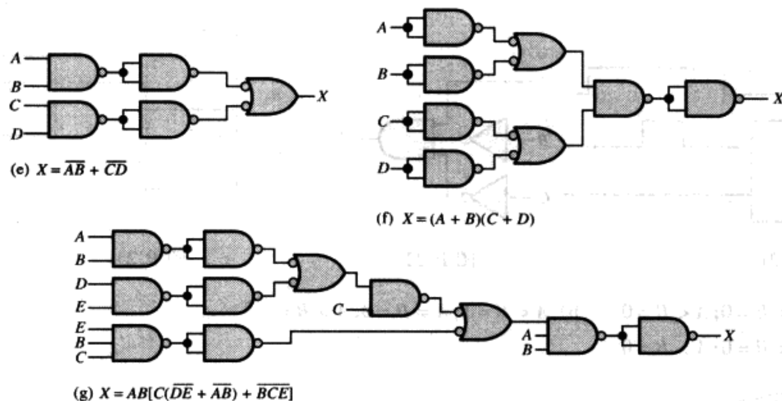


图 P.19

25. 参见图 P.20

27. $X = A + \overline{B}$;参见图 P.21

29. $X = A\overline{B}\overline{C}$;参见图 P.22

31. 输出脉冲宽度比规定的最小值大

第 6 章 组合逻辑电路函数

1. (a) $A \oplus B = 0, \Sigma = 1, (A \oplus B) C_{in} = 0, AB = 1, C_{out} = 1$

   (b) $A \oplus B = 1, \Sigma = 0, (A \oplus B) C_{in} = 0, AB = 0, C_{out} = 1$

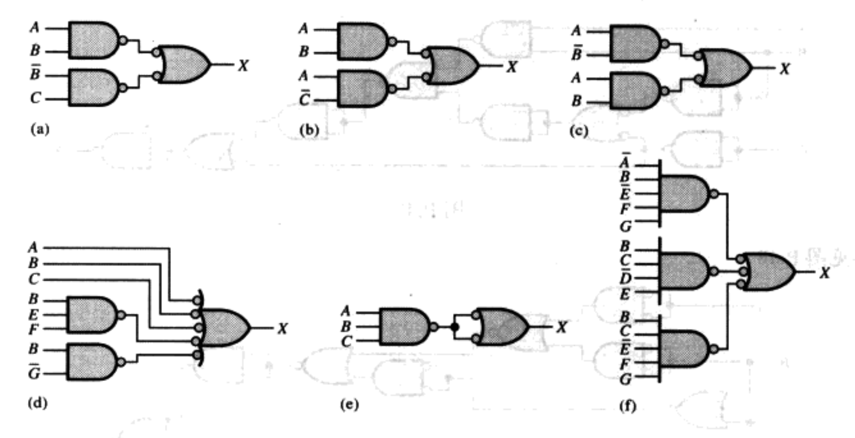


图 P.20

(c) $A \oplus B = 1, \Sigma = 1, (A \oplus B) C_{in} = 0, AB = 0, C_{out} = 0$

3. (a) $\Sigma = 1, C_{out} = 0$;　(b) $\Sigma = 1, C_{out} = 0$;　(c) $\Sigma = 0, C_{out} = 1$;　(d) $\Sigma = 1, C_{out} = 1$

5. 11100

7. $\Sigma_1 = 0110; \Sigma_2 = 1011; \Sigma_3 = 0110; \Sigma_4 = 0001; \Sigma_5 = 1000$

9. 225 ns

11. 当 $A_0 = B_0$ 和 $A_1 = B_1$ 时, $A = B$ 是高电平。参见图 P.23

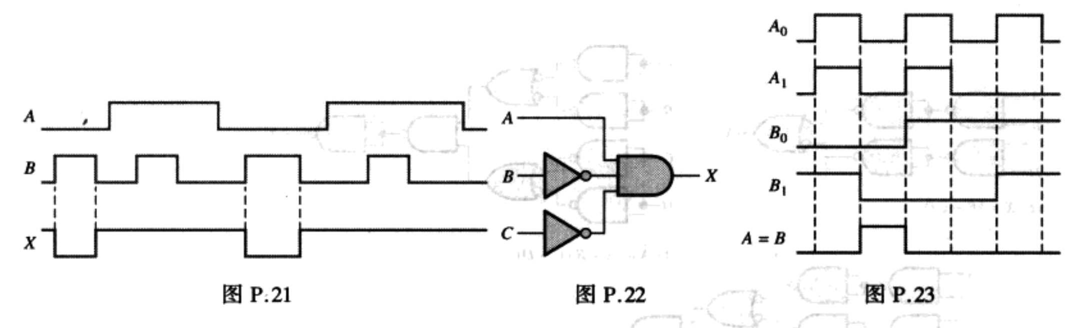


图 P.21　　图 P.22　　图 P.23

13. (a) $A > B = 1; A = B = 0; A < B = 0$　(b) $A < B = 1; A = B = 0; A > B = 0$

(c) $A = B = 1; A < B = 0; A > B = 0$

15. 参见图 P.24

17. $X = A_3 A_2 \overline{A}_1 \overline{A}_0 + \overline{A}_3 \overline{A}_2 \overline{A}_1 A_0 + A_3 \overline{A}_2 A_1$

19. 参见图 P.25

21. $A_3 A_2 A_1 A_0 = 1011$, 无效的 *BCD*

23. (a) $2 = 0010 = 0010_2$　(b) $8 = 1000 = 1000_2$　(c) $13 = 00010011 = 1101_2$

(d) $26 = 00100110 = 11010_2$ (e) $33 = 00110011 = 100001_2$

25. (a) 1010000000 格雷 → 1100000000 二进制

(b) 0011001100 格雷→ 0010001000 二进制

(c) 1111000111 格雷 → 1010000101 二进制

(d) 0000000001 格雷 → 0000000001 二进制

参见图 P.26

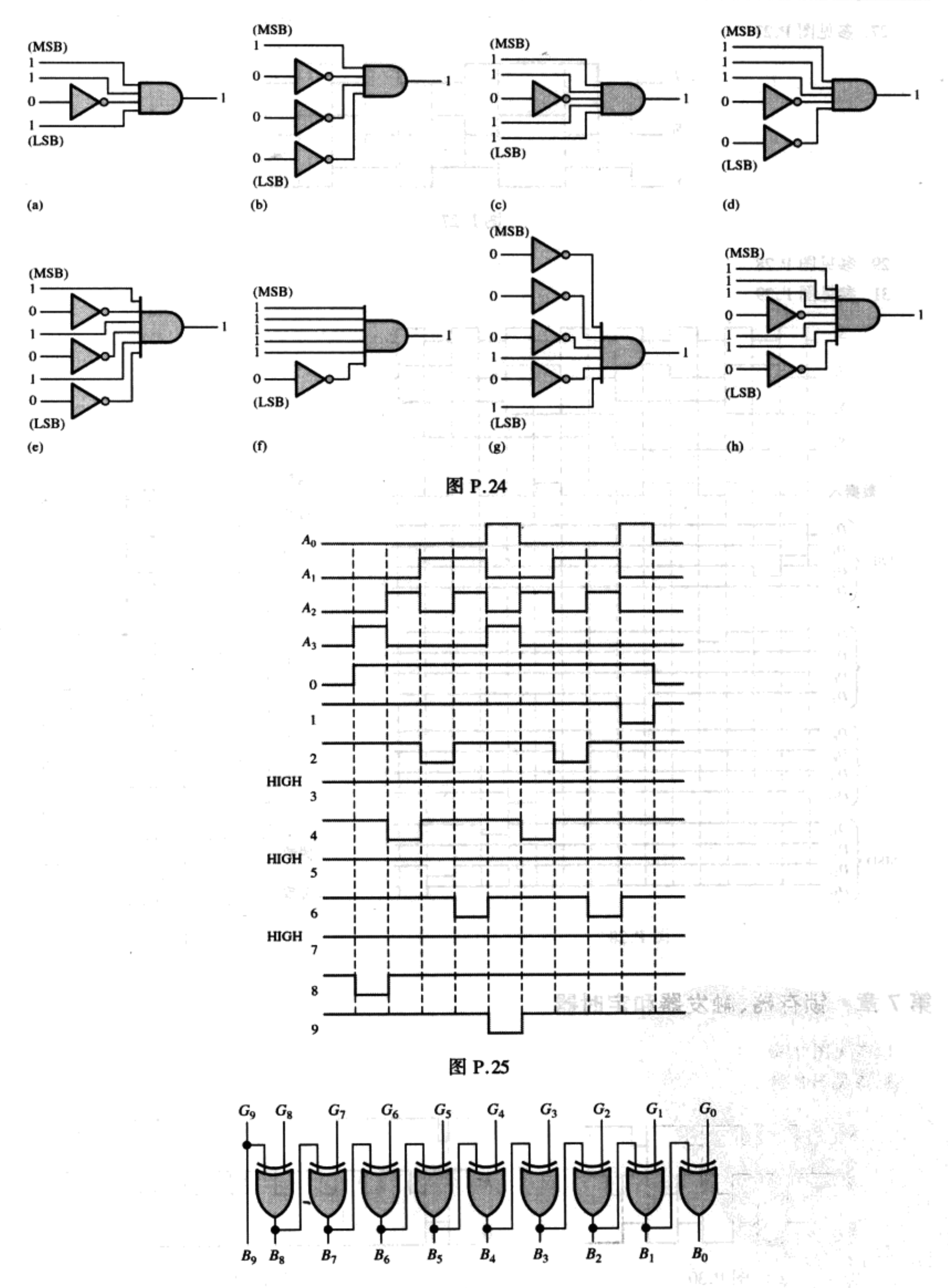


图 P.24

图 P.25

图 P.26

27. 参见图 P.27

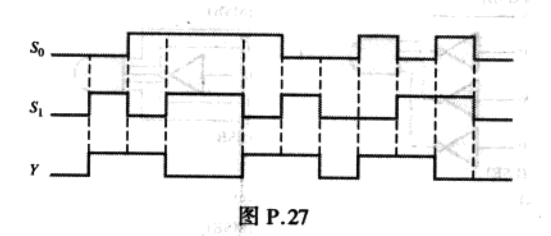


图 P.27

29. 参见图 P.28

31. 参见图 P.29

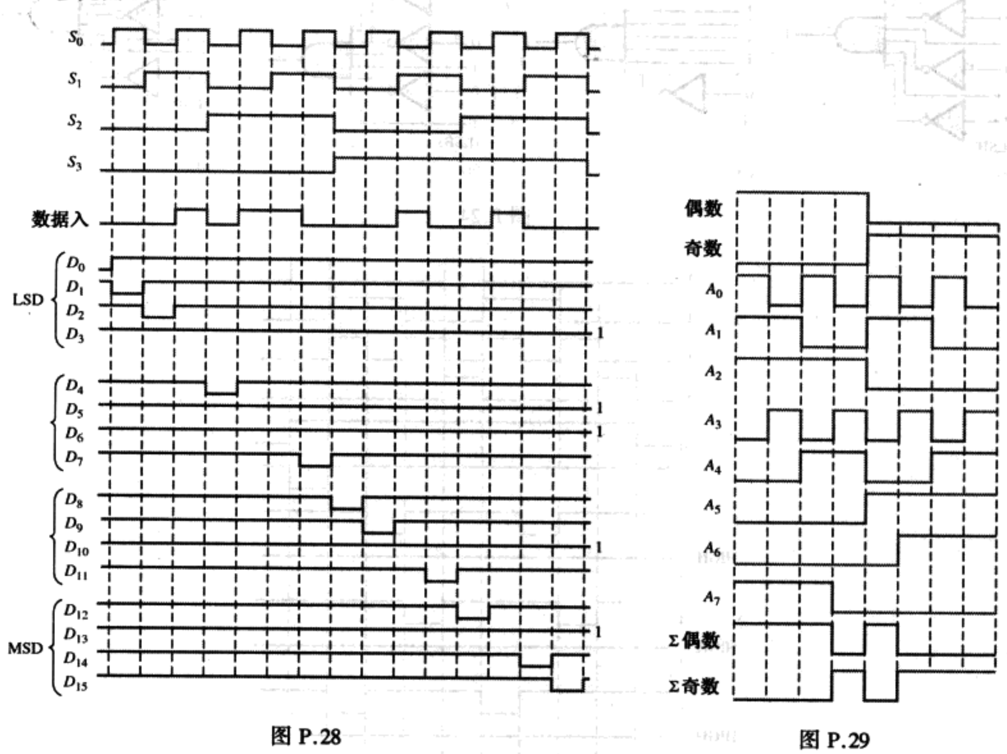


图 P.28　　　图 P.29

第 7 章　锁存器、触发器和定时器

1. 参见图 P.30

3. 参见图 P.31

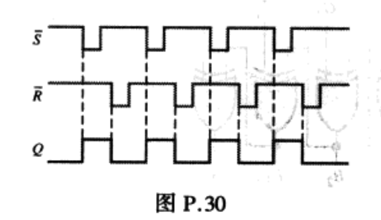


图 P.30

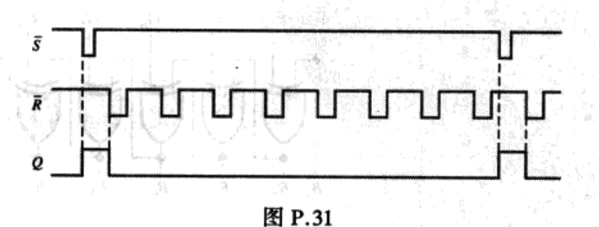
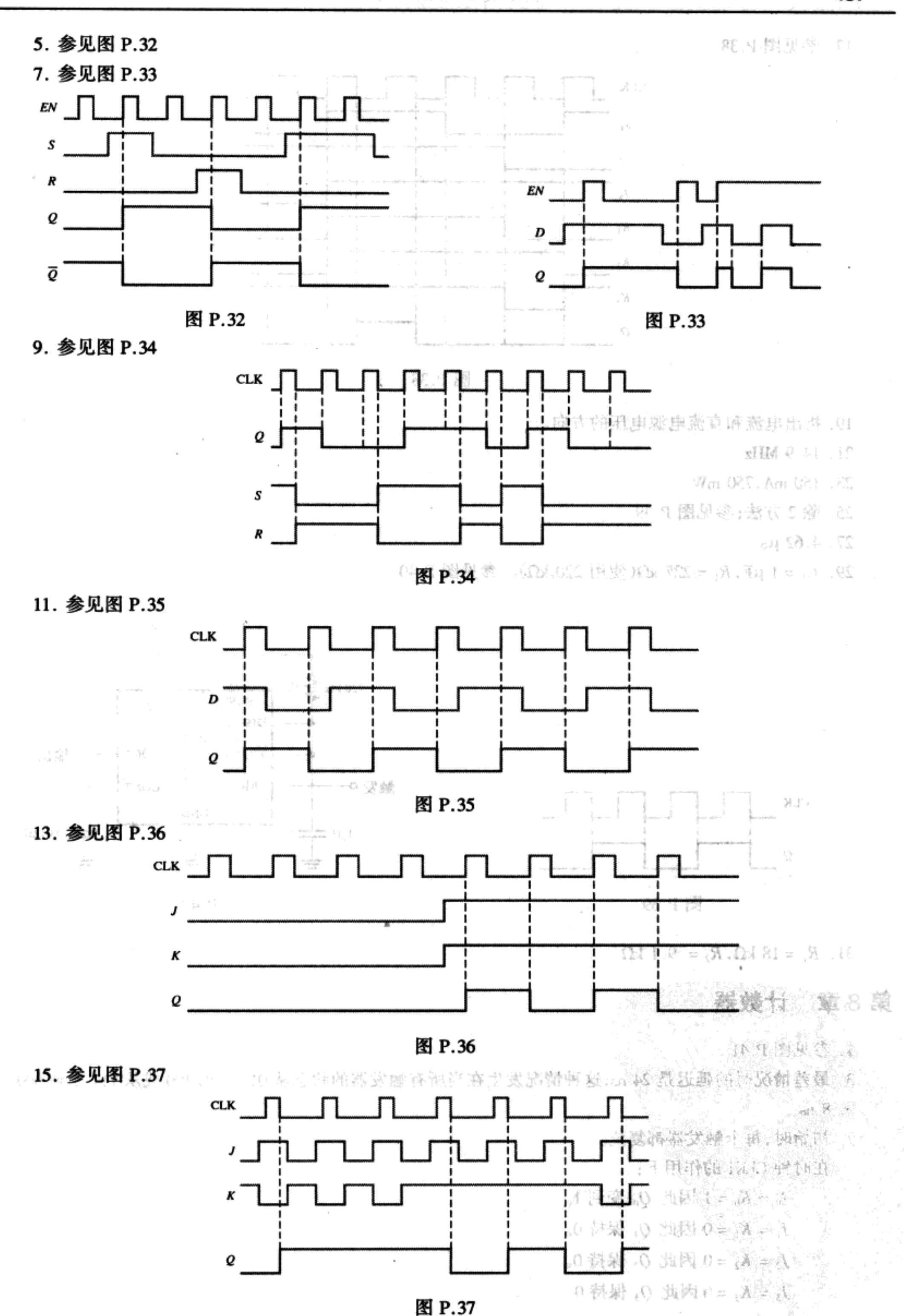


图 P.31



5. 参见图 P.32

7. 参见图 P.33

图 P.32

图 P.33

9. 参见图 P.34

图 P.34

11. 参见图 P.35

图 P.35

13. 参见图 P.36

图 P.36

15. 参见图 P.37

图 P.37

17. 参见图 P.38

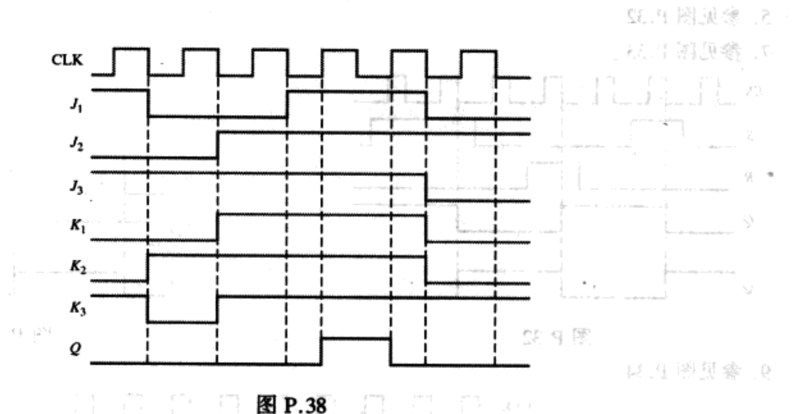


图 P.38

19. 指出电流和直流电源电压的方向。

21. 14.9 MHz

23. 150 mA,750 mW

25. 除 2 方法;参见图 P.39

27. 4.62 μs

29. $C_1 = 1\ \mu F$, $R_1 = 227\ k\Omega$(使用 220 kΩ)。参见图 P.40

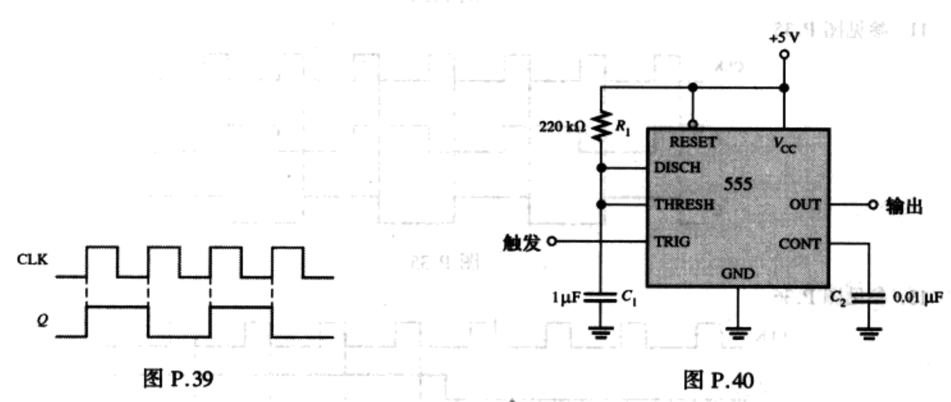


图 P.39

图 P.40

31. $R_1 = 18\ k\Omega$, $R_2 = 9.1\ k\Omega$

第 8 章 计数器

1. 参见图 P.41

3. 最差情况时的延迟是 24 ns;这种情况发生在当所有触发器的状态从 011 变到 100 或从 111 变到 000。

5. 8 ns

7. 初始时,每个触发器都复位。

   在时钟 CLK1 的作用下:

   $J_0 = K_0 = 1$ 因此 Q_0 变到 1。

   $J_1 = K_1 = 0$ 因此 Q_1 保持 0。

   $J_2 = K_2 = 0$ 因此 Q_2 保持 0。

   $J_3 = K_3 = 0$ 因此 Q_3 保持 0。

在时钟 CLK2 的作用下：

$J_0 = K_0 = 1$ 因此 Q_0 变到 0。

$J_1 = K_1 = 1$ 因此 Q_1 变到 1。

$J_2 = K_2 = 0$ 因此 Q_2 保持 0。

$J_3 = K_3 = 0$ 因此 Q_3 保持 0。

在时钟 CLK3 的作用下：

$J_0 = K_0 = 1$ 因此 Q_0 变到 1。

$J_1 = K_1 = 0$ 因此 Q_1 保持 1。

$J_2 = K_2 = 0$ 因此 Q_2 保持 0。

$J_3 = K_3 = 0$ 因此 Q_3 保持 0。

接下来的七个时钟脉冲，这个过程继续，也就指出计数器按照 BCD 码的序列计数。

9. 参见图 P.42

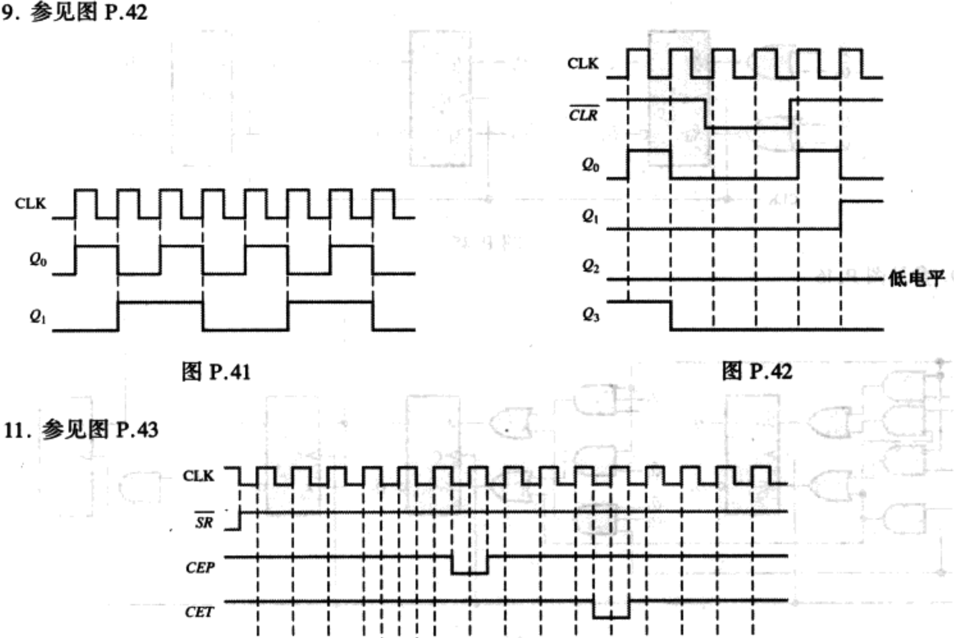


图 P.41　　图 P.42

11. 参见图 P.43

图 P.43

13. 参见图 P.44

15. 序列为 0000, 1111, 1110, 1101, 1010, 0101。计数器“自锁”在 1010 和 0101 状态，并在这两个数之间变化。

17. 参见图 P.45

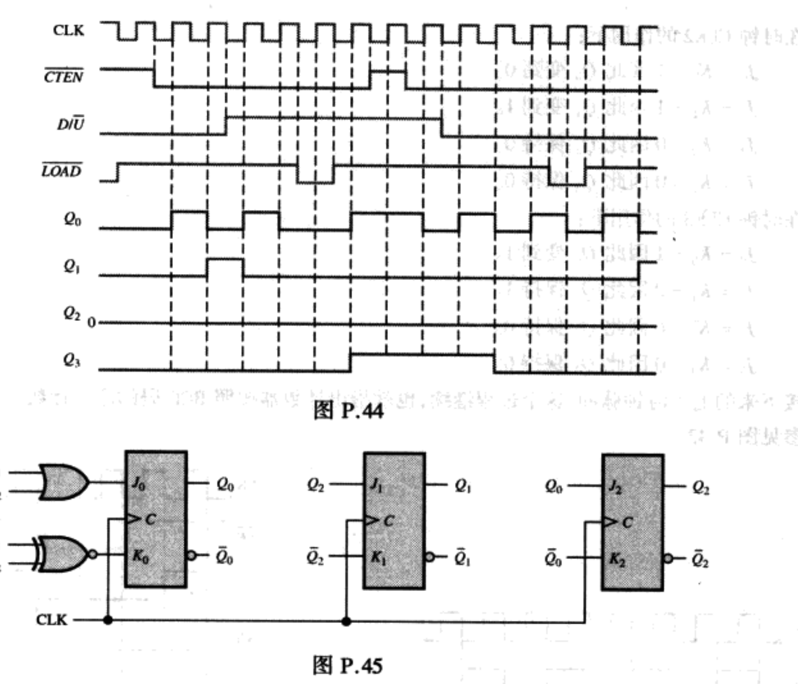


图 P.44

图 P.45

19. 参见图 P.46

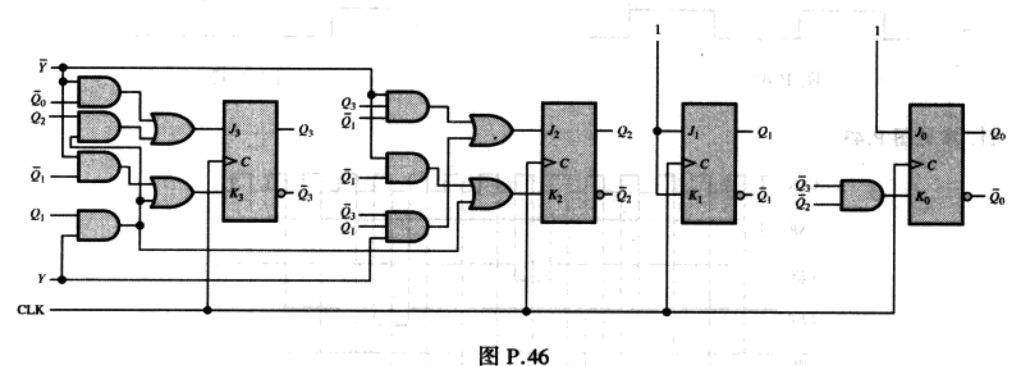


图 P.46

21. 参见图 P.47 的 10 000 分频电路。再添加一个除 10 计数器以产生 100 000 分频。

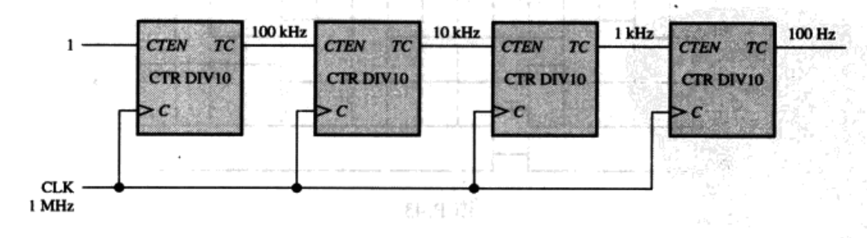


图 P.47

23. 参见图 P.48

25. CLK2,输出 0;CLK4,输出 2,0;CLK6,输出 4;CLK8,输出 6,4,0;CLK10,输出 8;CLK12,输出 10,8;CLK14,输出 12;CLK16,输出 14,12,8

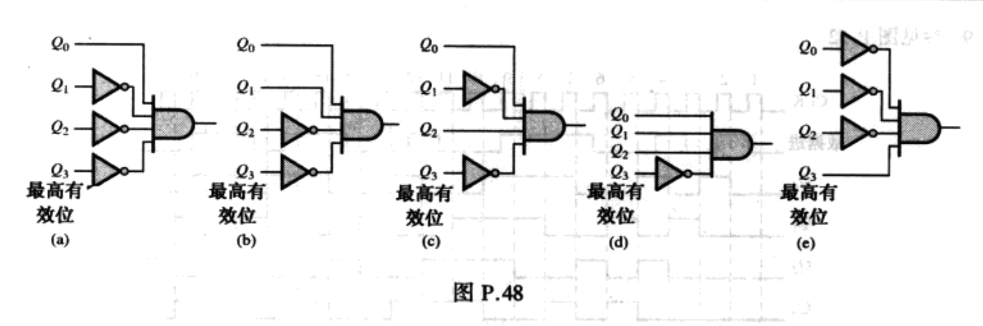


图 P.48

27. 与门输出的假信号发生在 111 到 000 的转换时间，通过 $\overline{CLK}$ 和计数器输出相与（选通）或使用格雷码消除假信号。

29. 参见图 P.49

31. 把 25 s 的单稳时间常数 $R_{EXT}C_{EXT}$ 增加 2.4 倍。

图 P.49

第 9 章　移位寄存器

1. 移位寄存器存储二进制数据
3. 参见图 P.50
5. 初始时：101001111000

   CLK1：010100111100

   CLK2：001010011110

   CLK3：000101001111

   CLK4：000010100111

   CLK5：100001010011

   CLK6：110000101001

   CLK7：111000010100

   CLK8：011100001010

   CLK9：001110000101

   CLK10：000111000010

   CLK11：100011100001

   CLK12：110001110000

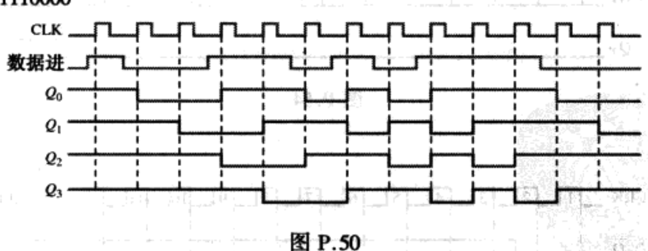


图 P.50

7. 参见图 P.51

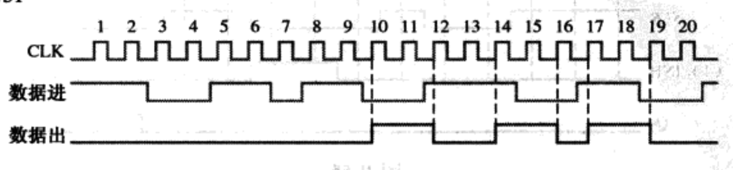


图 P.51

9. 参见图 P.52

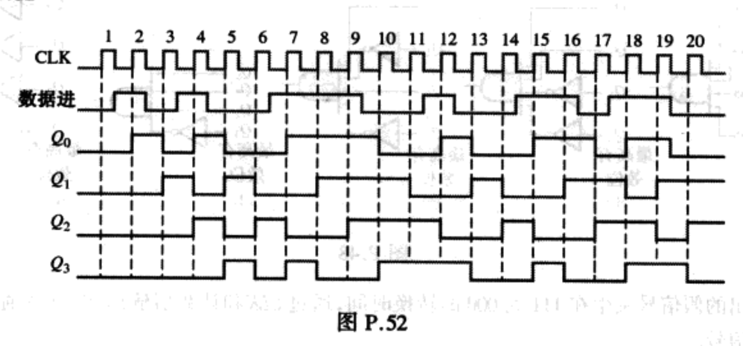


图 P.52

11. 参见图 P.53

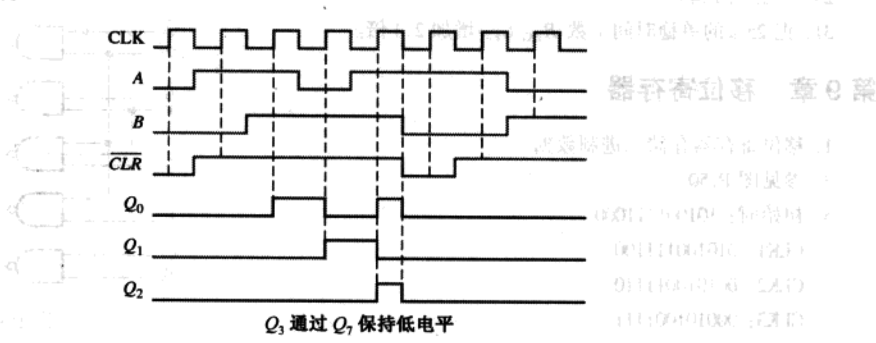


图 P.53

13. 参见图 P.54

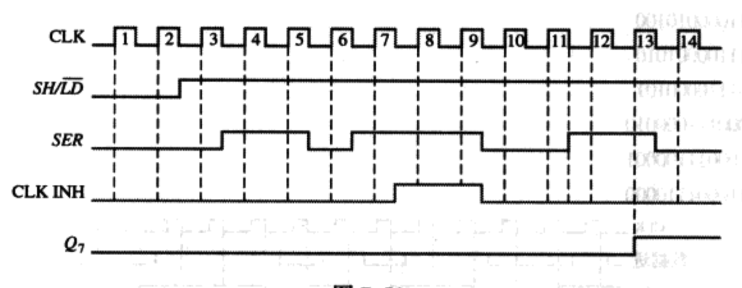


图 P.54

15. 参见图 P.55

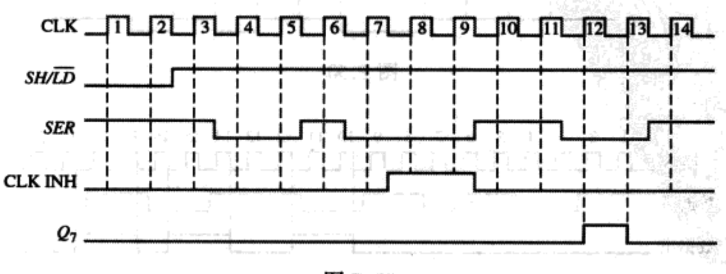


图 P.55

17. 参见图 P.56

19. 初始时（76）：01001100

CLK1：10011000 左

CLK2：01001100 右

CLK3：00100110 右

CLK4：00010011 右

CLK5：00100110 左

CLK6：01001100 左

CLK7：00100110 右

CLK8：01001100 左

CLK9：00100110 右

CLK10：01001100 左

CLK11：10011000 左

21. 参见图 P.57

23. (a) 3　(b) 5　(c) 7　(d) 8

25. 参见图 P.58

27. 参见图 P.59

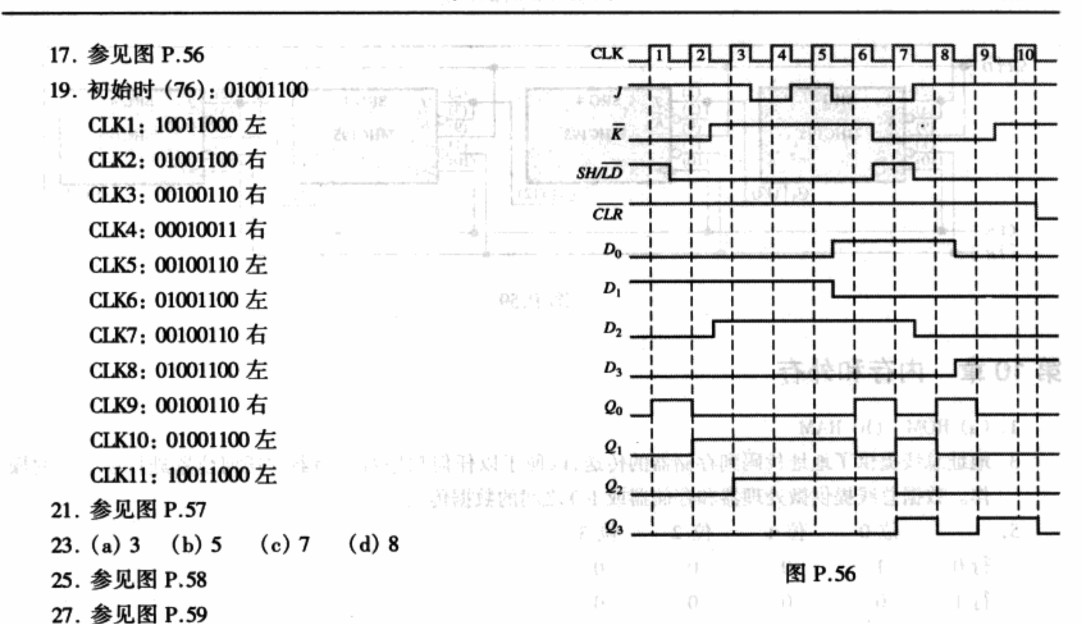


图 P.56

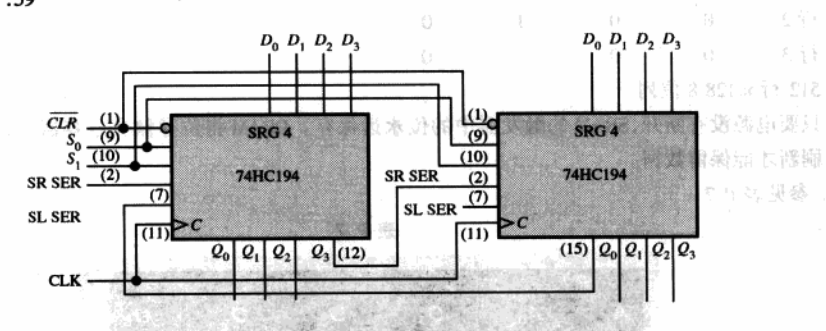


图 P.57

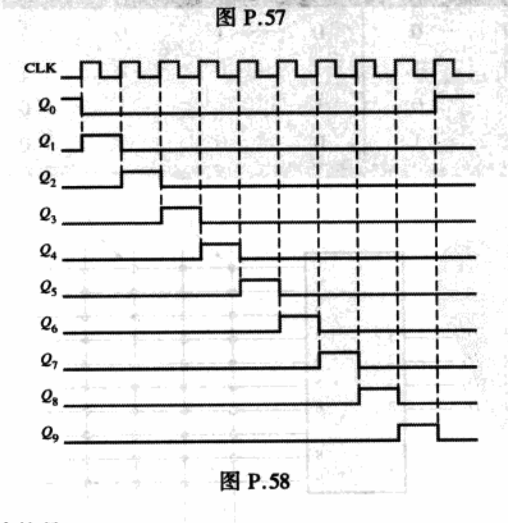


图 P.58

29. 可能产生一个不正确的码。

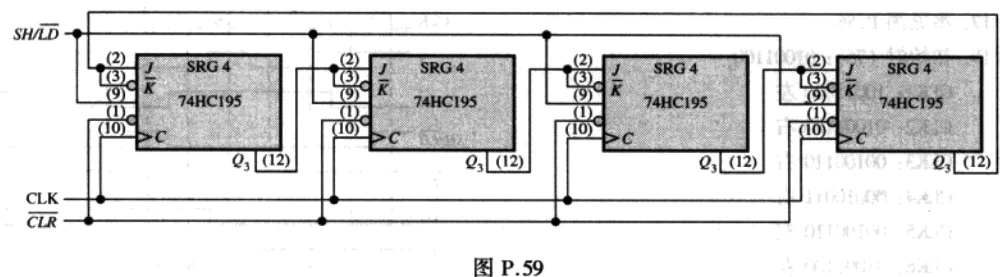


图 P.59

第 10 章 内存和外存

1. (a) ROM (b) RAM
3. 地址总线提供了地址代码到存储器的传送,以便于以任何顺序对存储器的任何位置进行读或写的操作。数据总线提供微处理器和存储器或 I/O 之间的数据传送。
5.

	位 0	位 1	位 2	位 3
行 0	1	0	0	0
行 1	0	0	0	0
行 2	0	0	1	0
行 3	0	0	0	0

7. 512 行 × 128 8 位列
9. 只要电源没有断开,SRAM 的触发器中的位永远保存。DRAM 将位存储在电容器中,但必须周期性的刷新才能保留数据。
11. 参见表 P.7

表 P.7

输入		输出			
A_1	A_0	O_3	O_2	O_1	O_0
0	0	0	1	0	1
0	1	1	0	0	1
1	0	1	1	1	0
1	1	0	0	1	0

13. 参见图 P.60

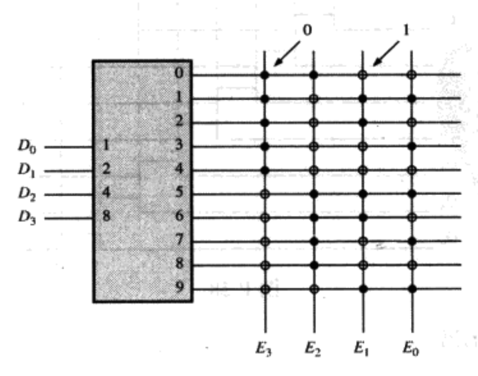


图 P.60

15. 并行可以实现同一时间执行 2 个指令。
17. 使用 16 条地址线的八个 16k×4 DRAM。其中两条地址线被译码用做存储器片选的使能输入。4 条数据线与每一个芯片相连接。
19. 8 位,64 k 字;4 位,256 k 字
21. 最低地址：$FC0_{16}$　最高地址：FFF_{16}

第 11 章　数字信号处理

1. 一个模数转换器将一个模拟信号转换成为数字码。
3. 一个数模转换器将一个数字代码转换成为相应的模拟信号。
5. 参见图 P.61

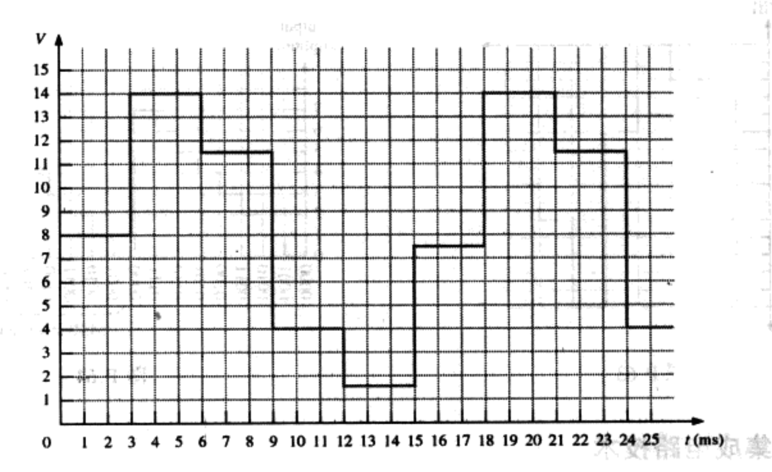


图 P.61

7. 1000,1110,1011,0100,0001,0111,1110,1011,0100
9. 参见图 P.62

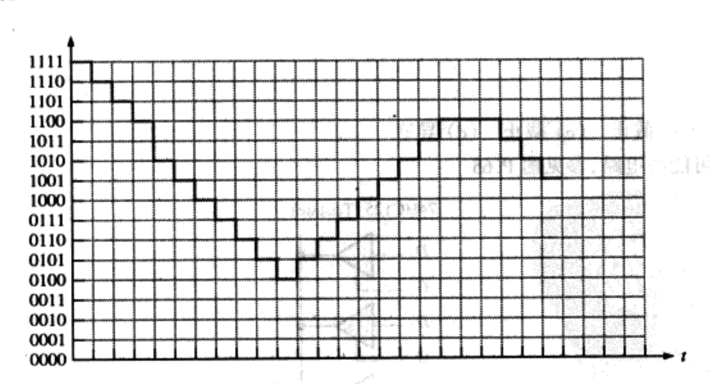


图 P.62

11. 330 kΩ
13. 000,001,100,110,101,100,011,010,001,001,011,110,111,111,111,111,111,111,111,100
15. 参见表 P.8

表 P.8

SAR	说明
1000	大于 V_{in}，复位 MSB
0100	小于 V_{in}，保持 1
0110	等于 V_{in}，保持（最后的状态）

18. 参见图 P.63

20. (a) 14.3%　(b) 0.098%　(c) 0.00038%

22. 参见图 P.64

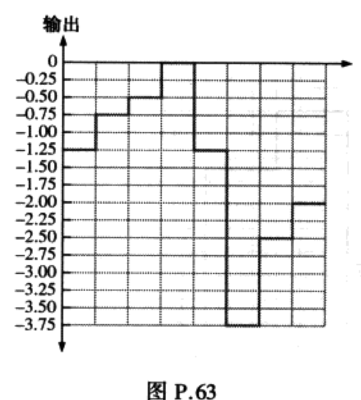


图 P.63

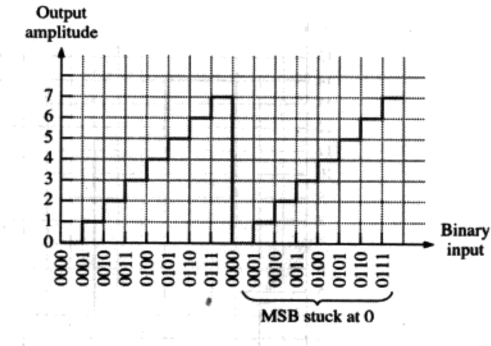


图 P.64

第 12 章　集成电路技术

1. 无; $V_{OH(min)} < V_{IH(min)}$

3. 0.15 V 处在高电平状态;0.25 V 处在低电平状态

5. 门 C

7. 12 ns

9. 门 C

11. 是, G_2

13. (a) 导通　(b) 截止　(c) 截止　(d) 导通

15. 对于一个可能的电路,参见图 P.65

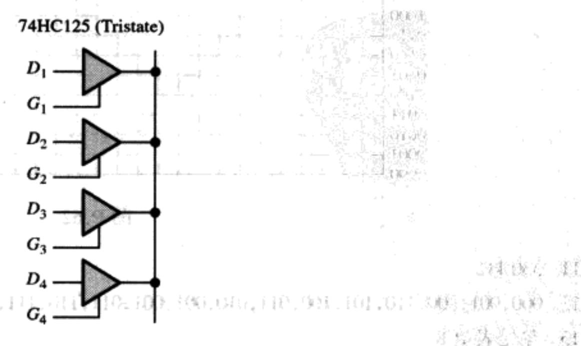


图 P.65

17. (a) 高电平 (b) 悬浮 (c) 高电平 (d) 高阻

19. (a) 低电平 (b) 低电平 (c) 低电平

21. 参见图 P.66

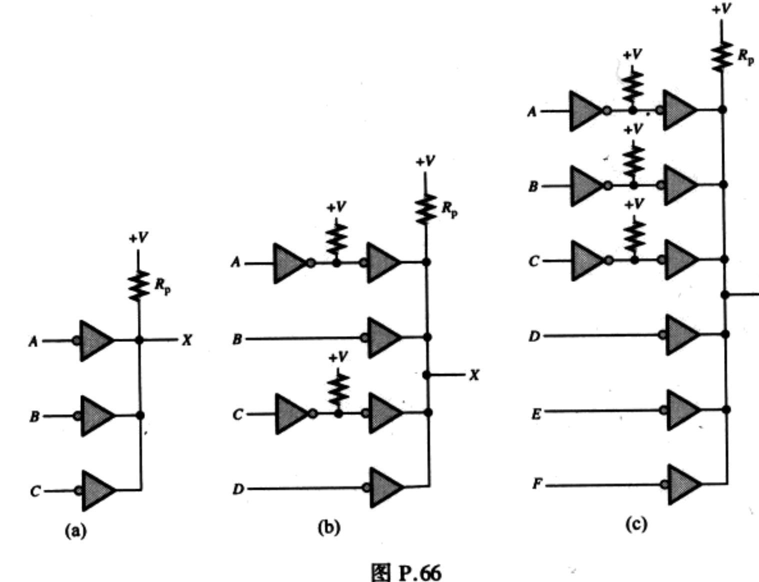


图 P.66

23. (a) $R_p = 198\ \Omega$ (b) $R_p = 198\ \Omega$ (c) $R_p = 198\ \Omega$

25. ALVC

27. (a) A,B 到 X:9.9 ns

C,D 到 X: 6.6 ns

(b) A 到 X_1, X_2, X_3: 14 ns

B 到 X_1: 7 ns

C 到 X_2: 7 ns

D 到 X_3:7 ns

(c) A 到 X: 11.1 ns

B 到 X: 11.1 ns

C 到 X: 7.4 ns

D 到 X: 7.4 ns

29. ECL 是 BJT 工作在非饱和状态下。